



# *HomeSeer® User's Guide*

***Version 1.6***

© 2002 HomeSeer Technologies LLC  
109 Powder Hill Rd  
Bedford NH 03110  
Phone 603-471-2816 Fax 603-471-9128  
E-mail: [support@homeseer.com](mailto:support@homeseer.com)  
Rev a



# Contents

<b>1.</b>	<b><u>Getting Started</u></b>	<b>9</b>
1.1	<u>New for version 1.6</u>	11
1.2	<u>Quick Start</u>	11
1.3	<u>HomeSeer Features</u>	12
1.4	<u>System Requirements</u>	13
1.5	<u>Technical Support</u>	13
1.6	<u>Running HomeSeer for the first time</u>	14
1.7	<u>Setting up Sunrise and Sunset times</u>	15
1.8	<u>HomeSeer Assistant Wizard</u>	15
1.9	<u>Importing data from ActiveHome or Home Director</u>	15
1.10	<u>Setting up EMAIL</u>	15
1.11	<u>Views</u>	17
1.12	<u>Device View</u>	17
1.13	<u>Events View</u>	18
1.14	<u>Event Log</u>	18
1.15	<u>Disabling Different types of Log Entries</u>	19
1.16	<u>Enable Event Logging to File</u>	19
1.17	<u>Enable Logging to Event View</u>	19
1.18	<u>Max size for Event Log (memory only)</u>	19
1.19	<u>Toolbar</u>	20
1.20	<u>Controlling your devices via email</u>	21
<b>2</b>	<b><u>Working with Devices</u></b>	<b>23</b>
2.1	<u>Supported Devices</u>	23
2.2	<u>Creating/Editing devices</u>	23
2.3	<u>Creating/Editing an X10 device</u>	26
2.4	<u>More on Virtual Devices</u>	26
2.5	<u>Devices that Represent Input/Output Points and Variables</u>	27
2.6	<u>Changing the status display</u>	28
2.7	<u>Creating Custom Device Types</u>	29
2.8	<u>Deleting Devices</u>	29
<b>3</b>	<b><u>Working with Events</u></b>	<b>31</b>
3.1	<u>Events Overview</u>	31
3.2	<u>Creating an event for a single device</u>	31
3.3	<u>Creating an event that controls multiple devices</u>	31
3.4	<u>Deleting Events</u>	33
3.5	<u>Enabling an Event as a Voice Command</u>	33
3.6	<u>Event Triggers</u>	34
3.7	<u>Manual Event Trigger</u>	37
3.8	<u>Recurring Event Trigger</u>	37
3.9	<u>Trigger Event by Absolute Time</u>	37
3.10	<u>Trigger Event by X10 Command</u>	39
3.11	<u>Trigger Event by Status Change</u>	40
3.12	<u>Trigger Event by Sunrise/Sunset</u>	40
3.13	<u>Trigger Event by condition</u>	41
3.14	<u>How Event Conditions Work</u>	42
3.15	<u>Trigger Event by EMAIL reception</u>	44
<b>4</b>	<b><u>Event Actions</u></b>	<b>45</b>
4.1	<u>Event Action Order</u>	45

<a href="#">4.2</a>	<a href="#">Device Specific Actions</a>	45	
<a href="#">4.3</a>	<a href="#">Global Actions</a>	48	
<a href="#">4.4</a>	<a href="#">Sending housecode wide X10 commands</a>	48	
<a href="#">4.5</a>	<a href="#">Launching an Application</a>	49	
<a href="#">4.6</a>	<a href="#">Playing a Sound</a>	49	
<a href="#">4.7</a>	<a href="#">Sending email</a>	49	
<a href="#">4.8</a>	<a href="#">Dialing an Internet Connection</a>	50	
<a href="#">4.9</a>	<a href="#">Running a script or speaking a phrase</a>	50	
<a href="#">4.10</a>	<a href="#">Triggering other Events</a>	53	
<a href="#">4.11</a>	<a href="#">Sending an infrared command</a>	53	
<a href="#">4.12</a>	<a href="#">Infrared Match Trigger</a>	54	
<a href="#">4.13</a>	<a href="#">Security Panel Trigger</a>	54	
<a href="#">4.14</a>	<a href="#">Security Panel Actions</a>	54	
<a href="#">4.15</a>	<a href="#">Timers</a>	54	
<a href="#">4.16</a>	<a href="#">Groups</a>	55	
<a href="#">4.17</a>	<a href="#">Security</a>	55	
<b>5</b>	<b><a href="#">Text to Speech Support and Voice Recognition</a></b>		<b>57</b>
<a href="#">5.1</a>	<a href="#">Voice Recognition</a>	57	
<a href="#">5.2</a>	<a href="#">Attention Phrases</a>	58	
<a href="#">5.3</a>	<a href="#">Intercepting the Attention Phrase</a>	59	
<a href="#">5.4</a>	<a href="#">Formatting Voice Commands</a>	60	
<a href="#">5.5</a>	<a href="#">New SAPI5 changes to voice command formats</a>	61	
<a href="#">5.6</a>	<a href="#">Using Dictation with Voice Recognition</a>	61	
<a href="#">5.7</a>	<a href="#">Creating Voice Commands to control devices</a>	62	
<a href="#">5.8</a>	<a href="#">Troubleshooting Speech Recognition</a>	62	
<a href="#">5.9</a>	<a href="#">Using MSAgent</a>	62	
<b>6</b>	<b><a href="#">Infrared Control Overview</a></b>		<b>64</b>
<a href="#">6.1</a>	<a href="#">PCIR/Linc</a>	65	
<a href="#">6.2</a>	<a href="#">Applied Digital CPU-XA/Ocelot</a>	65	
<a href="#">6.3</a>	<a href="#">JDS IRXpander</a>	65	
<a href="#">6.4</a>	<a href="#">PCRemote/PCIR</a>	66	
<a href="#">6.5</a>	<a href="#">Slink-e and HouseLinc</a>	66	
<a href="#">6.6</a>	<a href="#">Learning I/R Commands from Existing Remotes</a>	66	
<a href="#">6.7</a>	<a href="#">Changing I/R Device Key Labels</a>	67	
<a href="#">6.8</a>	<a href="#">Creating Voice Commands to Control I/R Devices</a>	68	
<b>7</b>	<b><a href="#">RF Control Overview</a></b>		<b>70</b>
<a href="#">7.1</a>	<a href="#">Setting up the MR26A RF Interface</a>	70	
<a href="#">7.2</a>	<a href="#">Setting up Events using the MR26A</a>	71	
<b>8</b>	<b><a href="#">Web Access</a></b>		<b>72</b>
<a href="#">8.1</a>	<a href="#">Starting the Web Server</a>	72	
<a href="#">8.2</a>	<a href="#">Using the HomeSeer Server with a Router</a>	73	
<a href="#">8.3</a>	<a href="#">Device Status Web Page</a>	74	
<a href="#">8.4</a>	<a href="#">Event List Web Page</a>	77	
<a href="#">8.5</a>	<a href="#">Creating your own Home Page</a>	78	
<a href="#">8.6</a>	<a href="#">Controlling devices and events from your own pages</a>	78	
<a href="#">8.7</a>	<a href="#">Controlling HomeSeer using ASP pages</a>	79	
<a href="#">8.8</a>	<a href="#">View only Web Access</a>	81	
<a href="#">8.9</a>	<a href="#">No Password for Subnet Access</a>	81	
<a href="#">8.10</a>	<a href="#">Customizing your HomeSeer Web Pages</a>	81	

<a href="#"><u>8.11</u></a>	<a href="#"><u>Remote File System Access</u></a>	84	
<b>9</b>	<b><a href="#"><u>Security Panel Support</u></a></b>		<b>86</b>
<a href="#"><u>9.1</u></a>	<a href="#"><u>Setting up the Napco interface</u></a>	86	
<b>10</b>	<b><a href="#"><u>HVAC Support</u></a></b>		<b>88</b>
<a href="#"><u>10.1</u></a>	<a href="#"><u>Setting up thermostats</u></a>	88	
<a href="#"><u>10.2</u></a>	<a href="#"><u>Controlling a thermostat with events</u></a>	89	
<a href="#"><u>10.3</u></a>	<a href="#"><u>Creating new scripts for other thermostats</u></a>	90	
<b>11</b>	<b><a href="#"><u>Power Failure support</u></a></b>		<b>92</b>
<a href="#"><u>11.1</u></a>	<a href="#"><u>Enabling Power Failure Support</u></a>	92	
<a href="#"><u>11.2</u></a>	<a href="#"><u>Disabling Power Failure Support on an Event/Device basis</u></a>	92	
<b>12</b>	<b><a href="#"><u>Printing</u></a></b>		<b>94</b>
<a href="#"><u>12.1</u></a>	<a href="#"><u>Printing event lists and controller labels</u></a>	94	
<b>13</b>	<b><a href="#"><u>TV/Video Game Timer</u></a></b>		<b>96</b>
<a href="#"><u>13.1</u></a>	<a href="#"><u>Setting up the TV/Video Game timer</u></a>	96	
<b>14</b>	<b><a href="#"><u>Options Summary</u></a></b>		<b>98</b>
<a href="#"><u>14.1</u></a>	<a href="#"><u>Options General</u></a>	98	
<a href="#"><u>14.2</u></a>	<a href="#"><u>Options Sunrise/Sunset</u></a>	99	
<a href="#"><u>14.3</u></a>	<a href="#"><u>Options EMAIL Setup</u></a>	100	
<a href="#"><u>14.4</u></a>	<a href="#"><u>Options Web Server</u></a>	101	
<a href="#"><u>14.5</u></a>	<a href="#"><u>Options Voice Recognition</u></a>	102	
<a href="#"><u>14.6</u></a>	<a href="#"><u>Options Text to Speech (TTS)</u></a>	102	
<a href="#"><u>14.7</u></a>	<a href="#"><u>Options Power Fail</u></a>	103	
<b>15</b>	<b><a href="#"><u>Scripting</u></a></b>		<b>104</b>
<a href="#"><u>15.1</u></a>	<a href="#"><u>Creating a Script</u></a>	104	
<a href="#"><u>15.2</u></a>	<a href="#"><u>Executing single script statements</u></a>	104	
<a href="#"><u>15.3</u></a>	<a href="#"><u>Debugging Scripts</u></a>	105	
<a href="#"><u>15.4</u></a>	<a href="#"><u>User supported scripts</u></a>	105	
<a href="#"><u>15.5</u></a>	<a href="#"><u>Common scripting questions</u></a>	106	
<a href="#"><u>15.6</u></a>	<a href="#"><u>Writing and reading from files using VBScript</u></a>	106	
<a href="#"><u>15.7</u></a>	<a href="#"><u>HomeSeer Scripting Interface (function quick reference)</u></a>	107	
<a href="#"><u>15.8</u></a>	<a href="#"><u>Function descriptions</u></a>	114	
<a href="#"><u>15.8.1</u></a>	<a href="#"><u>Misc Functions</u></a>	114	
<a href="#"><u>15.8.1.1</u></a>	<a href="#"><u>CreateVar</u></a>	114	
<a href="#"><u>15.8.1.2</u></a>	<a href="#"><u>DeleteVar</u></a>	114	
<a href="#"><u>15.8.1.3</u></a>	<a href="#"><u>SaveVar</u></a>	115	
<a href="#"><u>15.8.1.4</u></a>	<a href="#"><u>GetVar</u></a>	115	
<a href="#"><u>15.8.1.5</u></a>	<a href="#"><u>ClearLog</u></a>	116	
<a href="#"><u>15.8.1.6</u></a>	<a href="#"><u>SetVolume</u></a>	116	
<a href="#"><u>15.8.1.7</u></a>	<a href="#"><u>GetVolume</u></a>	116	
<a href="#"><u>15.8.1.8</u></a>	<a href="#"><u>version</u></a>	117	
<a href="#"><u>15.8.1.9</u></a>	<a href="#"><u>WaitSecs</u></a>	117	
<a href="#"><u>15.8.1.10</u></a>	<a href="#"><u>GetURL</u></a>	117	
<a href="#"><u>15.8.1.11</u></a>	<a href="#"><u>GetURLIE</u></a>	118	
<a href="#"><u>15.8.1.12</u></a>	<a href="#"><u>ftp</u></a>	118	

<a href="#">15.8.1.13</a>	<a href="#">FTPLastError</a>	120
<a href="#">15.8.1.14</a>	<a href="#">ftpSetProxy</a>	120
<a href="#">15.8.1.15</a>	<a href="#">SetRemoteTimeout</a>	121
<a href="#">15.8.1.16</a>	<a href="#">GetAppPath</a>	121
<a href="#">15.8.1.17</a>	<a href="#">GetIPAddress</a>	122
<a href="#">15.8.1.18</a>	<a href="#">Keys</a>	122
<a href="#">15.8.1.19</a>	<a href="#">Launch</a>	124
<a href="#">15.8.1.20</a>	<a href="#">Run</a>	125
<a href="#">15.8.1.21</a>	<a href="#">RunEx</a>	125
<a href="#">15.8.1.22</a>	<a href="#">StringItem</a>	127
<a href="#">15.8.1.23</a>	<a href="#">Sunrise</a>	127
<a href="#">15.8.1.24</a>	<a href="#">Sunset</a>	127
<a href="#">15.8.1.25</a>	<a href="#">SetSecurityMode</a>	128
<a href="#">15.8.1.26</a>	<a href="#">WriteLog</a>	128
<a href="#">15.8.1.27</a>	<a href="#">WaitEvents</a>	128
<a href="#">15.8.1.28</a>	<a href="#">X10InterfaceStatus</a>	129
<a href="#">15.8.1.29</a>	<a href="#">WEBStatsPageViews</a>	130
<a href="#">15.8.1.30</a>	<a href="#">WEBLoggedInUser</a>	130
<a href="#">15.8.1.31</a>	<a href="#">WEBValidateUser</a>	131
<a href="#">15.8.1.32</a>	<a href="#">Ping</a>	131
<a href="#">15.8.1.33</a>	<a href="#">System</a>	131
<a href="#">15.8.1.34</a>	<a href="#">SystemUptime</a>	131
<a href="#">15.8.1.35</a>	<a href="#">Shutdown</a>	132
<a href="#">15.8.1.36</a>	<a href="#">PlayWavFile</a>	132
<a href="#">15.8.1.37</a>	<a href="#">PlayWavFileEx</a>	132
<a href="#">15.8.1.38</a>	<a href="#">ScheduleFile</a>	133
<a href="#">15.8.1.39</a>	<a href="#">GetINISection</a>	133
<a href="#">15.8.1.40</a>	<a href="#">GetINISetting</a>	134
<a href="#">15.8.1.41</a>	<a href="#">ClearINISection</a>	135
<a href="#">15.8.1.42</a>	<a href="#">SaveINISetting</a>	135
<a href="#">15.8.1.43</a>	<a href="#">GetPlugins</a>	135
<a href="#">15.8.1.44</a>	<a href="#">plugin</a>	136
<a href="#">15.8.1.45</a>	<a href="#">GetPrinter</a>	136
<a href="#">15.8.1.46</a>	<a href="#">PrintOut</a>	137
<a href="#">15.8.1.47</a>	<a href="#">IsScriptRunning</a>	137
<a href="#">15.8.1.48</a>	<a href="#">NoLog</a>	137
<a href="#">15.8.1.49</a>	<a href="#">ControlThermostat</a>	138
<a href="#">15.8.2</a>	<a href="#">Media Functions</a>	139
<a href="#">15.8.2.1</a>	<a href="#">MediaMute</a>	139
<a href="#">15.8.2.2</a>	<a href="#">MediaPlay</a>	139
<a href="#">15.8.2.3</a>	<a href="#">MediaVolume</a>	139
<a href="#">15.8.2.4</a>	<a href="#">MediaPause</a>	140
<a href="#">15.8.2.5</a>	<a href="#">MEDIAFilename</a>	140
<a href="#">15.8.2.6</a>	<a href="#">MediaIsPlaying</a>	140
<a href="#">15.8.2.7</a>	<a href="#">MediaStop</a>	141
<a href="#">15.8.3</a>	<a href="#">Communications Port Functions (RS232)</a>	142
<a href="#">15.8.3.1</a>	<a href="#">OpenComPort</a>	142
<a href="#">15.8.3.2</a>	<a href="#">CloseComPort</a>	144
<a href="#">15.8.3.3</a>	<a href="#">GetComPortCount</a>	144
<a href="#">15.8.3.4</a>	<a href="#">GetComPortData</a>	144
<a href="#">15.8.3.5</a>	<a href="#">SendToComPort</a>	145
<a href="#">15.8.3.6</a>	<a href="#">SetComPortRTSDTR</a>	145
<a href="#">15.8.4</a>	<a href="#">Infrared Functions</a>	146
<a href="#">15.8.4.1</a>	<a href="#">SendIR</a>	146
<a href="#">15.8.5</a>	<a href="#">Voice Related Functions</a>	147
<a href="#">15.8.5.1</a>	<a href="#">ClearAllVoiceCommands</a>	147
<a href="#">15.8.5.2</a>	<a href="#">LastVoiceCommand</a>	147

---

<a href="#">15.8.5.3</a>	<a href="#">LastCommandSelected</a>	148
<a href="#">15.8.5.4</a>	<a href="#">AddVoiceCommand</a>	148
<a href="#">15.8.5.5</a>	<a href="#">ListenForCommands</a>	150
<a href="#">15.8.5.6</a>	<a href="#">SetSpeaker</a>	150
<a href="#">15.8.5.7</a>	<a href="#">StartListen</a>	150
<a href="#">15.8.5.8</a>	<a href="#">StopListen</a>	151
<a href="#">15.8.5.9</a>	<a href="#">Speak</a>	151
<a href="#">15.8.5.10</a>	<a href="#">StopSpeaking</a>	152
<a href="#">15.8.5.11</a>	<a href="#">SpeakToFile</a>	152
<a href="#">15.8.5.12</a>	<a href="#">SpeakEx</a>	154
<a href="#">15.8.5.13</a>	<a href="#">MuteSpeech</a>	154
<a href="#">15.8.5.14</a>	<a href="#">ListenMode</a>	155
<a href="#">15.8.6</a>	<a href="#">Device Functions</a>	156
<a href="#">15.8.6.1</a>	<a href="#">DeviceStringByName</a>	156
<a href="#">15.8.6.2</a>	<a href="#">SetDeviceStringByName</a>	156
<a href="#">15.8.6.3</a>	<a href="#">NewDevice</a>	156
<a href="#">15.8.6.4</a>	<a href="#">NewDeviceEx</a>	157
<a href="#">15.8.6.5</a>	<a href="#">GetDevice</a>	157
<a href="#">15.8.6.6</a>	<a href="#">GetDeviceEx</a>	159
<a href="#">15.8.6.7</a>	<a href="#">DeviceCount</a>	159
<a href="#">15.8.6.8</a>	<a href="#">GetDeviceCode</a>	160
<a href="#">15.8.6.9</a>	<a href="#">PollDevice</a>	160
<a href="#">15.8.6.10</a>	<a href="#">ExecX10ByName</a>	161
<a href="#">15.8.6.11</a>	<a href="#">ExecX10</a>	162
<a href="#">15.8.6.12</a>	<a href="#">IsOnByName</a>	163
<a href="#">15.8.6.13</a>	<a href="#">IsOffByName</a>	164
<a href="#">15.8.6.14</a>	<a href="#">IsOff</a>	165
<a href="#">15.8.6.15</a>	<a href="#">IsOn</a>	166
<a href="#">15.8.6.16</a>	<a href="#">DeviceStatus</a>	168
<a href="#">15.8.6.17</a>	<a href="#">SetDeviceStatus</a>	168
<a href="#">15.8.6.18</a>	<a href="#">DeviceValue</a>	168
<a href="#">15.8.6.19</a>	<a href="#">SetDeviceValue</a>	170
<a href="#">15.8.6.20</a>	<a href="#">SetDeviceString</a>	171
<a href="#">15.8.6.21</a>	<a href="#">DeviceString</a>	171
<a href="#">15.8.6.22</a>	<a href="#">DeviceValueByName</a>	173
<a href="#">15.8.6.23</a>	<a href="#">SetDeviceValueByName</a>	174
<a href="#">15.8.6.24</a>	<a href="#">DeviceTime</a>	175
<a href="#">15.8.6.25</a>	<a href="#">SetDeviceTime</a>	175
<a href="#">15.8.6.26</a>	<a href="#">DeviceLastChange</a>	175
<a href="#">15.8.6.27</a>	<a href="#">SetDeviceLastChange</a>	177
<a href="#">15.8.6.28</a>	<a href="#">DeviceExists</a>	177
<a href="#">15.8.6.29</a>	<a href="#">GetDeviceList</a>	178
<a href="#">15.8.6.30</a>	<a href="#">GetDeviceByRef</a>	178
<a href="#">15.8.6.31</a>	<a href="#">LastX10</a>	178
<a href="#">15.8.6.32</a>	<a href="#">ClearLastX10</a>	179
<a href="#">15.8.6.33</a>	<a href="#">DeviceTimeByName</a>	180
<a href="#">15.8.6.34</a>	<a href="#">RegisterStatusChangeCB</a>	180
<a href="#">15.8.6.35</a>	<a href="#">UnRegisterStatusChangeCB</a>	181
<a href="#">15.8.6.36</a>	<a href="#">DeviceButtonAdd</a>	182
<a href="#">15.8.6.37</a>	<a href="#">DeviceButtonRemove</a>	184
<a href="#">15.8.7</a>	<a href="#">Event Functions</a>	186
<a href="#">15.8.7.1</a>	<a href="#">GetLastEvent</a>	186
<a href="#">15.8.7.2</a>	<a href="#">TriggerEvent</a>	186
<a href="#">15.8.7.3</a>	<a href="#">DeleteEvent</a>	187
<a href="#">15.8.7.4</a>	<a href="#">DisableEvent</a>	187
<a href="#">15.8.7.5</a>	<a href="#">EnableEvent</a>	187
<a href="#">15.8.7.6</a>	<a href="#">NewTimeEvent</a>	188

<a href="#">15.8.7.7</a>	<a href="#">NewRecurringEvent</a>	189
<a href="#">15.8.7.8</a>	<a href="#">DelayTrigger</a>	190
<a href="#">15.8.7.9</a>	<a href="#">EventCount</a>	190
<a href="#">15.8.7.10</a>	<a href="#">GetEvent</a>	190
<a href="#">15.8.7.11</a>	<a href="#">GetEventEx</a>	196
<a href="#">15.8.7.12</a>	<a href="#">GetEventList</a>	197
<a href="#">15.8.7.13</a>	<a href="#">EventExists</a>	197
<a href="#">15.8.7.14</a>	<a href="#">NewEvent</a>	197
<a href="#">15.8.7.15</a>	<a href="#">NewCondition</a>	198
<a href="#">15.8.7.16</a>	<a href="#">RemoveDelayedEvent</a>	199
<a href="#">15.8.7.17</a>	<a href="#">NewDevFunc</a>	199
<a href="#">15.8.7.18</a>	<a href="#">SaveEventsDevices</a>	201
<a href="#">15.8.8</a>	<a href="#">EMAIL Functions</a>	202
<a href="#">15.8.8.1</a>	<a href="#">SendEmail</a>	202
<a href="#">15.8.8.2</a>	<a href="#">MailMsgCount</a>	202
<a href="#">15.8.8.3</a>	<a href="#">MailFrom</a>	202
<a href="#">15.8.8.4</a>	<a href="#">MailFromDisplay</a>	202
<a href="#">15.8.8.5</a>	<a href="#">MailDate</a>	203
<a href="#">15.8.8.6</a>	<a href="#">MailSubject</a>	203
<a href="#">15.8.8.7</a>	<a href="#">MailText</a>	203
<a href="#">15.8.8.8</a>	<a href="#">MailTo</a>	203
<a href="#">15.8.8.9</a>	<a href="#">MailToDisplay</a>	204
<a href="#">15.8.8.10</a>	<a href="#">MailDelete</a>	204
<a href="#">15.8.8.11</a>	<a href="#">MailTrigger</a>	204
<a href="#">15.8.9</a>	<a href="#">CPU-XA/Ocelot Functions</a>	205
<a href="#">15.8.9.1</a>	<a href="#">cpuxa.GetPoint</a>	205
<a href="#">15.8.9.2</a>	<a href="#">cpuxa.LearnIRStart</a>	205
<a href="#">15.8.9.3</a>	<a href="#">cpuxa.LearnIREnd</a>	206
<a href="#">15.8.9.4</a>	<a href="#">cpuxa.PlayIR</a>	206
<a href="#">15.8.9.5</a>	<a href="#">cpuxa.SendRaw</a>	207
<a href="#">15.8.9.6</a>	<a href="#">cpuxa.SetVar</a>	207
<a href="#">15.8.9.7</a>	<a href="#">cpuxa.SetPoint</a>	207
<a href="#">15.8.9.8</a>	<a href="#">cpuxa.PlayRemotelR</a>	208
<a href="#">15.8.10</a>	<a href="#">System Functions</a>	209

## [HomeSeer as an ActiveX Server](#)

210

## [16 Index](#)

213



---

# Chapter 1

---

## 1. Getting Started

### Welcome to HomeSeer!

This user's guide is for use with version 1.6 of the HomeSeer software.

Please see `readme.txt` file for a list of changes since version 1.5 was released.

If you want to dive right into the software, please take a quick look at this quick start guide.

HomeSeer is an advanced home automation package designed to take full advantage of the features available in your PC such as Internet access, email, text-to-speech, and others. These features are typically not available in stand-alone controllers. The software was designed to be easy to use, yet flexible enough for advanced users.

HomeSeer is designed to work with a wide range of home automation devices. Support for new devices is through the use of *Plug-Ins*. New plug-ins are constantly becoming available through HomeSeer Technologies, as well as third parties and end users. Check our support web page and our message board for the latest plug-ins that are available. Some plug-ins are purchased separately. Here is a partial list of hardware that HomeSeer supports:

- X10 ActiveHome CM11A X10 serial interface (two way interface)
- X10 CM17A RF serial interface (one way, send only X10 interface)
- All CM11A compatible interfaces such as the IBM Home Director HD11A, the Radio Shack plug'n power interface, and the X10 CM12U.
- Applied Digital Ocelot/CPU-XA (X10/Infrared/Input-output serial interface)
- MR26A RF Receiver
- ACT TI103 X10 interface (X10 only)
- SmartLinc HouseLinc home controller (X10 and infrared)
- NIRVIS Slink-e (infrared, Sony Control-S)
- JDS IRXpander (Infrared)
- JDS Stargate
- Custom Solutions HomeVision
- SmartHome PowerLinc
- SmartHome IRLinc model 1623PC
- HAI Omni series Security Controllers
- RCS TX15B and TR15, Enerzone (Aprilaire), HAI RC80 thermostats
- Web Cameras
- Windows Media Player
- Support for other hardware is available on our website at [www.homeseer.com/support.htm](http://www.homeseer.com/support.htm)

Although the CM11A is packaged with its own software, HomeSeer offers many more features and is much easier to use. You will need additional hardware to utilize HomeSeer.

The software is designed to work with just about any X10 compatible device. If you are confused about what Home Automation or X10 is, please read the FAQ section before going any further. The FAQ is available at the HomeSeer Technologies web site at [www.homeseer.com](http://www.homeseer.com).

At a minimum, you will need the following:

- CM11A or TI103 X10 interface. These controllers connect to your PC's serial port.
- Some X10 plug in lamp modules
- Some X10 plug in appliance modules
- At least one table top X10 controller (such as the X10 Powerhouse Mini Controller)

The procedure for using HomeSeer is to first create all your devices, then create events to control these devices. See the sections on adding devices and adding events for more information.

**Some key information:**

- HomeSeer has an *autosave* feature. Any changes you make to your configuration, either locally or via the web, are saved immediately. There is no need to be concerned about losing data and no need to perform frequent saves. It may be wise to backup your configuration file periodically. Note that HomeSeer automatically creates a backup file of your configuration. Configuration files have the extension ".xml". All configuration information is stored in the *config* directory in the HomeSeer folder.
- The current configuration file is always displayed in the title bar of the main window.
- You can easily view all the events associated with a particular device. Simply click the device and all the associated events will appear in the bottom pane. You can add more events to this device by right clicking in this pane. If the bottom pane is not visible, right click in the devices pane and select *Show Events*
- Your configuration file is always backed up when HomeSeer starts. The file has the same name as your configuration file but has *bak* in the filename.
- HomeSeer has an auto update feature. To upgrade the application to the latest version, select Updates from the help menu in HomeSeer. The updater will also allow you to install special script packages, plug-ins, and other third party software such as the Napco security panel support and the CM11A download feature.

## New for version 1.6

Here is a list of the changes from 1.5 to 1.6.

- Added ASP support for the web server. This allows dynamic web pages to be created and served by the HomeSeer web server
- Ability to control devices using your voice, like "turn on the living room lights". No event needs to be created to do this.
- Better plug-in support and support for more third party hardware
- Better voice recognition. HomeSeer now supports the latest SAPI5 voice recognition and the new voice recognition engine provided by Microsoft. This is the same voice recognition that is supplied with Office XP. Note that support for SAPI4 voice recognition has been dropped, and support for using the MS Agent for voice recognition is no longer available. HomeSeer is now always in *Always Listen* mode. Dictation is now available for use in voice commands.
- Added support for SAPI5 text-to-speech. This includes support for the AT&T Natural Voices. The AT&T TTS is among the best available. Note that SAPI4 TTS is still supported as well as support for MS Agent. Note that the agent cannot speak using any SAPI5 voice.
- More flexible printing support.
- Plug-ins available for Video camera support and Media Player support.
- New XML file format for configuration files.
- Device status is saved and restored internally rather than by a script.
- Power failure recovery support. Devices and events can be selected to be restored in the event the power is disrupted. On startup, the selected devices state will be restored.
- Many more minor changes throughout the program.

### 1.1 Quick Start

This section is intended to get you started quickly using HomeSeer without reading the entire manual, although reading the manual is highly recommended. HomeSeer is loaded with features that may not be obvious by looking at the setup screens.

- **Tell HomeSeer about your hardware.** Click the Interfaces tab in the HomeSeer options from the menu View->Options. Remember to restart the application after making any interface changes.
- **Using the voice recognition**  
Enable the voice recognition on the *Voice Recognition* tab in the options. After closing the options dialog, press *Control-L* on the keyboard. This starts the computer listening. To get the computers attention, speak the attention phrase, which is *Computer* by default. When the computer acknowledges you, you may then speak a command. The event list will show all your events. The events that respond to voice commands have a "v" in the column with the "V" heading. Create voice commands by entering the command phrase in the Voice Command text box in the properties of an event.

To control individual devices using your voice, check the box "Auto Voice Command" in the properties for the device. You can then say "turn on/off the [device location name]", like "turn off the living room lights", or "living room lights off".

- **To see all the events associated with a device, click the device in the device view. All the events will appear in the bottom pane.**
- **Right click a device or event to display its attributes (properties)**
- **To organize your events, put them into different groups.**  
Events may be assigned a group name. Give related events the same group name. This makes

it easier to display related events. Groups may be displayed by selecting the proper group name from the drop down list in the events display.

- **If you have a CM11A and a CPU-XA/Ocelot, you can use the CM11A for X10 and Ocelot for IR and I/O.** Just give each device a different COM port. Note that the Ocelot cannot receive extended X10 commands properly.
- **Remember to enable listening with the menu Tools->Listen for commands**  
If you set the voice recognition to Always Listen mode, you will not have to hit the Scroll Lock key to speak. However, make sure you turn on listening by either pressing Control-L or by checking the menu item *Tools->Listen for commands*. Note that the status line will display the current listening mode.
- **Look at the sample events**  
The sample configuration file (sample.xml) contains some sample events that can help you create your own events.

## 1.2 HomeSeer Features

- Works with many popular X10 and infrared controllers.
- Works great on Windows XP, Windows 2000, Windows NT, Windows ME, or Windows 98. Windows 95 is not supported, it also strongly suggested that Windows ME not be used. The ideal operating system is Windows 2000 or Windows XP.
- Telephone support is available in the form of an add-on that allows HomeSeer to be controlled from any telephone. This support also adds answering machine and full caller-id support. See [www.homeseer.com](http://www.homeseer.com) for more information.
- Supports the NAPCO Gemini and HAI Omni security panels (extra cost add-on)
- Supports the X10 MR26A RF receiver, this enables HomeSeer to receive commands from X10 motion sensors, thereby speeding up motions sensor response.
- Built In web server allows full control of all your X10 devices via your home network or the internet
- Password secure log-in to web server
- If you have a MAPI compliant email client such as Outlook Express, HomeSeer can display your local email on a web page.
- Ability to control any X10 compatible device, including devices that support pre-set dim commands as well as Leviton devices that support the extended dim command.
- Security mode varies timed events randomly plus or minus 30 minutes.
- Runs scripts created in vbscript, jscript, or perlscript. (Only vbscript and jscript are included in the distribution)
- An extensive programming interface is provided for writing scripts that will totally automate the program.
- Includes Microsoft Agent. Microsoft Agents are interactive characters that can speak. HomeSeer can cause the Agent to speak in response to events. Events can also be triggered with voice commands. HomeSeer includes the "Genie" character. Other characters are freely available on the Internet.
- Voice control is available. Simply say "computer", and the program will listen for your commands.
- Supports infrared devices such as the SmartHome IRLinc model 1623PC, the CPU-XA/Ocelot, the ConceptUK PCRemote, JDS IRXpander, and Nirvis Slink-e device.
- Supports SmarLinc SwitchLinc, Compose and PCX wall switches.
- Supports the RCS TX15B, RCSTR15, Enerzone (Aprilaire), and HAI RC80 thermostats.
- Supports the JDS Stargate and Custom Solutions HomeVision controllers.
- More hardware support is available at [www.homeseer.com/support.htm](http://www.homeseer.com/support.htm)
- Device commands may be imbedded in email messages. This allows you to send an email home to turn devices on and off.
- Views are designed to show as many devices or events as possible on your screen, giving

- the user instant status of devices
- View devices by device or by event, displaying all events related to a device, or all devices related to an event
- Events may be triggered by time, X10 command received, sunrise/sunset, various conditions like elapsed time on or off, by email, by infrared match, by a device status change, by a phone event such as a ring or caller ID available (HomeSeer Phone software required) or a security panel event.
- Trigger actions include, controlling X10 devices, launching applications, sending EMAIL, playing sounds, sending I/R commands, triggering other events, speaking with or without MSAgent, dialing an internet connection, or controlling a security panel. If HomeSeer Phone is installed, phone actions are also available such as changing the number of rings before calls are answered.
- Built-in TV/Video game timer, which allows you to set time limits on the use of TV's or video games. Kids are given a 4-digit password that is entered on a tabletop X10 controller. HomeSeer will then turn off the TV after a pre-determined period of time has elapsed. Time may be set by day or by week.
- Full logging of all actions either performed locally or via the network.
- Microsoft Outlook user interface.
- A special "Voice Control" web page can be displayed with your custom voice commands.
- HomeSeer is implemented as an ActiveX EXE. It can be controlled by other applications including IIS web servers.

### 1.3 System Requirements

HomeSeer only runs under Windows XP/2000/NT4/ME or 98. It will not work under Windows 3.1, or DOS. You will need a Web browser installed if you would like to view your devices via a web page or view the online documentation. Windows 95 is not supported.

HomeSeer has been tested with Netscape and Microsoft Internet Explorer.

TCP/IP must be installed on your PC if you are going to use the built-in web server or email.

If you would like to view your local mail, a MAPI compliant email application should be installed, such as Outlook Express or Outlook 98/2000.

You will need an Internet account to access your system via the Web.

HomeSeer must be running in order for it to control your X10 devices. It will not download any timers or macros to the CM11A interface or the CPU-XA/Ocelot/JDS/HomeVision interface. A 3<sup>rd</sup> party add-on is available that does allow HomeSeer to download timers and macros to the CM11A interface. See the 3<sup>rd</sup> party folder on the CD or visit [www.hawizard.com](http://www.hawizard.com) for more information.

If you intend on using voice recognition, you should have a Pentium processor running at 300 mhz or faster. You also need a good quality microphone for the best recognition accuracy.

### 1.4 Technical Support

Support for HomeSeer is available via EMAIL at [support@homeseer.com](mailto:support@homeseer.com). All users may send support questions to this address. If you are a registered user, you will receive priority support. Please include the serial number from your version of HomeSeer. We also have a message board available on our web site. From the message board you can view questions and answers other users have asked and you can post your questions where others may benefit from the answers. The message board is available at:

<http://www.homeseer.com> (click on the message board link at the top of the page)

Product enhancement requests and general comments may be sent to [feedback@homeseer.com](mailto:feedback@homeseer.com) or visit our web site and submit your question to our message board.

Please visit [www.homeseer.com](http://www.homeseer.com) and check the support page for updates and answers to frequently asked questions. The HomeSeer message board is another excellent source for information.

Product updates are available from within HomeSeer. Select *Updates* from the Help menu to get the latest version of the software as well as scripts, plug-ins, and other third party software.

Our development process is very open. As new features are added to the software, updates will be available via the HomeSeer updater, including Beta versions. Keep in mind that Beta versions of the software may not be as stable as the released version.

## 1.5 Running HomeSeer for the first time

Connect up the CM11A as per its included instructions. Basically, you plug it in the wall, and then attach one end of the supplied serial cable to the module and the other end to any free serial port on your PC. Double click the setup application and follow the prompts to install the software. If you have the CD version of HomeSeer, the HomeSeer installer will start when you insert your CD into your CDROM drive.

A HomeSeer folder is added to your start menu. A HomeSeer icon is added to this folder. Select *Start* then *programs* then *HomeSeer*, and then select the HomeSeer icon. HomeSeer will launch and display a message about searching for the CM11A. If you do not have a CM11A yet, but wish to see the program in action, simply press the *Cancel Search* button. After a few seconds the dialog will disappear and the main window will appear. If you do have a CM11A connected, HomeSeer will search all your serial ports. If it displays an error about NOT finding it, try telling HomeSeer what port your CM11A is connected to using the following procedure.

- Enter the Options screen by selecting the menu *View->Options* and select the *Interfaces* tab.
- Select the proper X10 device and COM ports from the dropdown lists.
- Restart HomeSeer.
- When the program finishes loading, check the event log by clicking the *Log* button on the left side of the screen. If it still gives you an error, try the following:
  - Check all your connections
  - It's possible the CM11A is hung. The CM11A contains a microprocessor that seems to lock up at times. There is a simple way to fix it however. Using any tabletop keypad, press a couple of buttons to send some X10 commands. The CM11A will receive these commands and unlock itself. Now restart HomeSeer again and check your event log. This should unlock it.

**You may want to also click on the Clear CM11A Memory button to make sure there are no timers programmed into the CM11A. To clear the memory, click the "options" button on the interfaces tab in the HomeSeer options.**

HomeSeer will load a default configuration file named *sample.xml*. HomeSeer configuration files have the extension *.xml*. This file contains some sample devices and sample events. The file is located in the same directory as the HomeSeer.exe application.

If you want to view your devices from the built in Web Server, see *Starting the Web Server* for more information.

## 1.6 Setting up Sunrise and Sunset times

You can have HomeSeer calculate sunrise and sunset times for your location.

- Open the options dialog from the menu *View->Options->Sunrise/Sunset*
- Select your location from the drop down list: click OK.
- If your location is not listed, enter your longitude and latitude, or select the closest location. Enter longitude values as positive values east of longitude 0, and negative west of longitude 0.
- Click the *Calculate* button if you entered longitude and latitude values.

## 1.7 HomeSeer Assistant Wizard

To help you get started using HomeSeer quickly, you can use the *HomeSeer Assistant Wizard* to step you through some common tasks. The wizard is displayed by default when you first install HomeSeer. Simply follow the prompts. The wizard is also available from the *Tools* menu and the Toolbar.

## 1.8 Importing data from ActiveHome or Home Director

If you previously used ActiveHome or IBM's Home Director as your home automation software, you can have HomeSeer import information about your devices and events. Run the *HomeSeer Assistant Wizard* and select *Import* from ActiveHome. You will be prompted for your ActiveHome \*.X10 (Home Director \*.HOM) device file and \*.MAC macro file. These files are normally located in "c:\program files\home control\data".

### Note the following when importing data:

- The locations of the devices are not imported. This information is not available in either the .X10 or .MAC files. You will need to assign locations to your devices yourself
- Timed events that are valid only for a date range are imported properly as HomeSeer does not support date ranges for events, only absolute dates.

## 1.9 Setting up EMAIL

If you intend to use HomeSeer to send you email in response to events or use email to control your devices, then read this section on how to configure email.

HomeSeer can also display all the email messages that are in your inbox on a web page. **This feature only works if you are using a MAPI compliant email system.** Most email clients including Outlook Express and Outlook 98/2000 support MAPI. It also works with Netscape. You need to enable MAPI in the preferences for Netscape. Your messages will appear on the mail page of your web site. See the Web Access section for more information.

You need to tell HomeSeer some information about how to send email. In the menu *View->Options->Email* dialog enter the following information:

- 1 If you have a MAPI compatible email client such as Outlook Express or Outlook98/2000, check the checkbox *Use MAPI to handle email*. This will tell HomeSeer to use the MAPI interface for sending and receiving email. If you are going to use email as an event trigger, then you must tell HomeSeer how often to check for new email. Check the checkbox *check for mail every # minutes* and enter the number of minutes between checks. Note that HomeSeer only asks the

MAPI interface for new mail and does not check your actual email account at your internet service provider. You must set up your email client to actually do the email checking. HomeSeer also supports auto logging into an Exchange server. See the *Username* and *Password* fields below.

- 2 If you do not wish to use any of the email features, then leave the *smtp server* entry blank, uncheck the box labeled *Check for mail every # minutes* and uncheck the box labeled *Use MAPI to handle mail*.

If you wish to check and send email via your ISP's POP and SMTP email servers, then uncheck the *Use MAPI to handle email* checkbox and enter the following information:

**SMTP server**

This is the address of your SMTP server. This is the same server name you entered in the setup of your standard email program.

**Your EMAIL address**

This is your email address. It will appear in the *from* field of the email.

**EMAIL notifications are sent to this address**

Enter the email address of where you would like the email notifications to be sent. This could be your work email or may be a free web based email account.

**Default Subject**

Enter the text you would like to appear in the subject of all email notifications.

**Default Message**

If you would like some message to be displayed in the message field of all EMAIL notifications, enter it here. Note that HomeSeer will always include a message about what caused the notification to be sent. You can enter a different message for each event. You can do this when you create the event.

**POP server**

This is your ISP's pop server address.

**Username/Profile name**

The username you use to login to your ISP's POP server. If you are using MAPI, you may enter your profile name for logging into an Exchange server.

**Password**

The password you use to login to your ISP's POP server. If you are using MAPI, you may enter your password for logging into an Exchange server.

**Use MAPI to handle mail**

HomeSeer will use the MAPI mail interface to handle email. All email settings are controlled by your email client application such as Outlook Express or Outlook98

**Check for mail every # minutes**

HomeSeer will check for new email every number of minutes entered. This is used only for triggering events by email. If you are not using email to trigger events, then you can leave this unchecked.



## 1.10 Views

The leftmost pane is the View Bar. This pane contains buttons that allow you select what type of information you wish to display. The View Bar may be hidden by either right clicking on it and selecting *hide view bar* or by unchecking view bar from the *View* menu.

### See Also

Event Log  
Device View

## 1.11 Device View

This is a list of all your devices with their current status (On, Off, or Dim). The devices may be sorted by clicking the titles of any column. You may also limit the view to a particular location. Just below the menu bar there is a drop down list of all your locations (or *Rooms*). Select a location from the list, such as *Family Room*. The list will now display devices that are in that room. Select *All Locations* to get a list of all your devices. If you would like to see all the events associated with a particular device, first right click in the devices pane. This will bring up a pop-up menu. Click on *Show Events*. This will split the devices pane into two panes, the top pane will continue to show all your devices, while the bottom pane will show all events associated with a selected device. Simply click on a device and all events associated with this device will be displayed. You can hide this event pane by right clicking in the events pane and selecting *Hide Events* from the pop-up menu.

Note that you can re-order the columns by clicking and dragging the column headers.

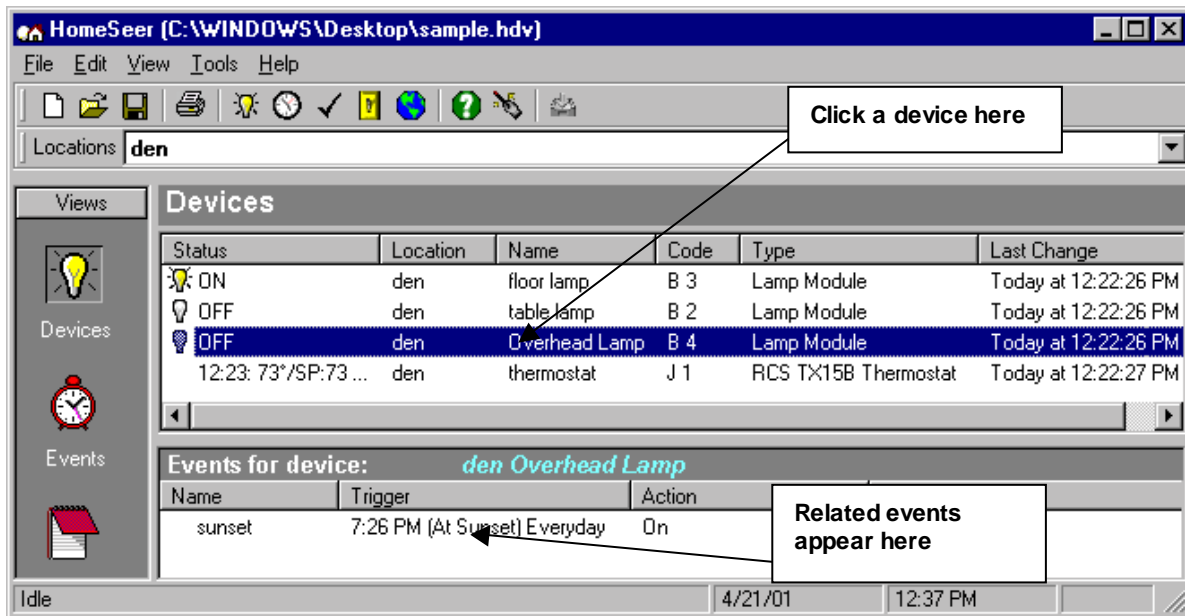


Figure 1 Devices View

## 1.12 Events View

Displays a list of all your configured events. Click on an event and all the devices that this event controls will be listed in the bottom pane. From the bottom pane, you can right click on a device and remove it from the event, edit its properties, edit its actions, or add another device. See the *Events* sections for more information on events. The view lists the following columns:

- **V** = the event is a voice command
- **S** = SmartOn is enabled for this event. Any X10 devices that are to be controlled will only be sent ON/DIM commands if they are OFF.
- **E** = the event will send an email message when triggered
- **Name** = the name of the event
- **Trigger** = the type of trigger for the event and will describe what must happen for the event to trigger
- **Events** = other events that will run when this event triggers
- **Scripts** = scripts that are executed when the event triggers
- **I/R Command** = the infrared commands that are sent when the event triggers
- **Global X10** = housecode wide X10 commands that are executed when the event triggers
- **Application** = the application that is launched when the event triggers
- **Sound** = the sound that is played when the event triggers
- **Speak** = the phrase that is spoken when the event triggers (text to speech must be enabled)
- **Group** = The name of the group the event belongs to

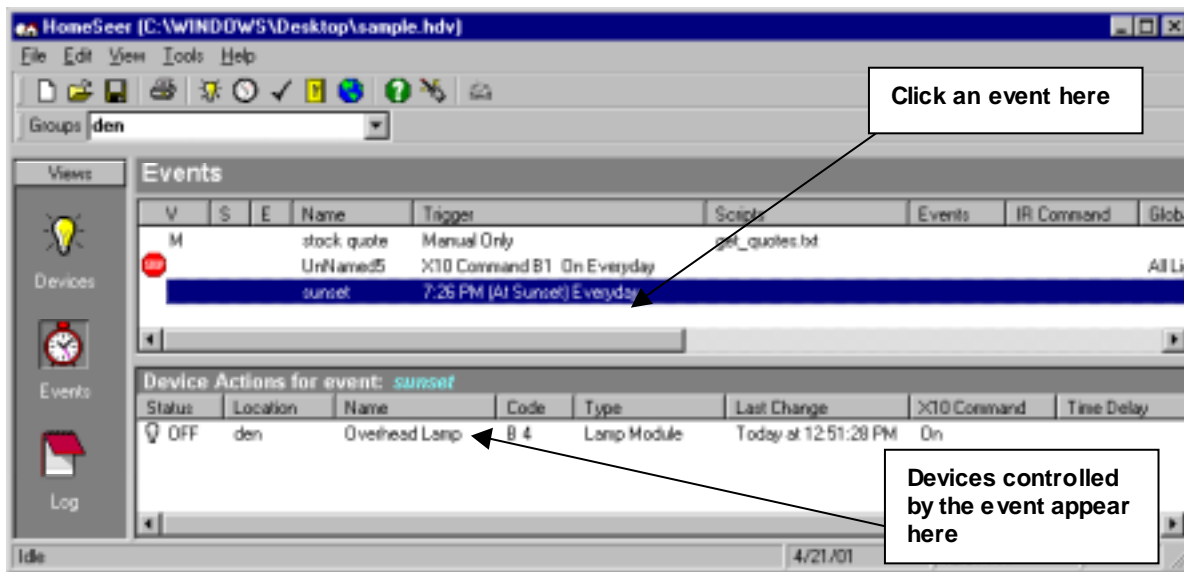


Figure 2 Events View

## 1.13 Event Log

The event log is used to keep track of HomeSeer activity. It logs all event activity as well as X10 activity on your power line. If you are troubleshooting some events that do not seem to be working properly, the log is an invaluable tool for seeing what's going on. You can configure the log from the Options dialog at *View->Options->General*. The options are:

The log screen displays a history of what actions HomeSeer has performed. The following is logged:

- All X10 signals seen on the power line
- All event triggers, no matter how they are triggered

- All logins to the built-in web server. The log includes the IP address of the host who logged in
- All device actions performed via the web interface
- All actions performed via the ActiveX (OLE) interface, this includes scripts.

### See Also

Disabling Different types of Log Entries  
Enable Event Logging to File

Enable Logging to Event View  
Max size for Event Log (memory only)

## 1.14 Disabling Different types of Log Entries

At times it may be desirable to disable the logging of certain actions. For example, you may have a motion sensor that is very active, and this is filling the log with entries. You can disable some logging as follows:

- To disable logging of device actions, check the *No X10 Logging* check box in the devices properties.
- To prevent an event from logging when it is triggered, check the *Do not Log this Event* check box in the events properties.
- To disable the logging of web server connection errors, uncheck the *Log Errors* check box on the Web Server tab in the program options.

## 1.15 Enable Event Logging to File

This option is available from the menu *View->Options->General*.

All log entries are written to the file *ah.log*. The log file is in the HomeSeer application directory. The file is never deleted automatically. You can delete the log file whenever you like. A new file is created if it does not exist.

## 1.16 Enable Logging to Event View

This option is available from the menu *View->Options->General*.

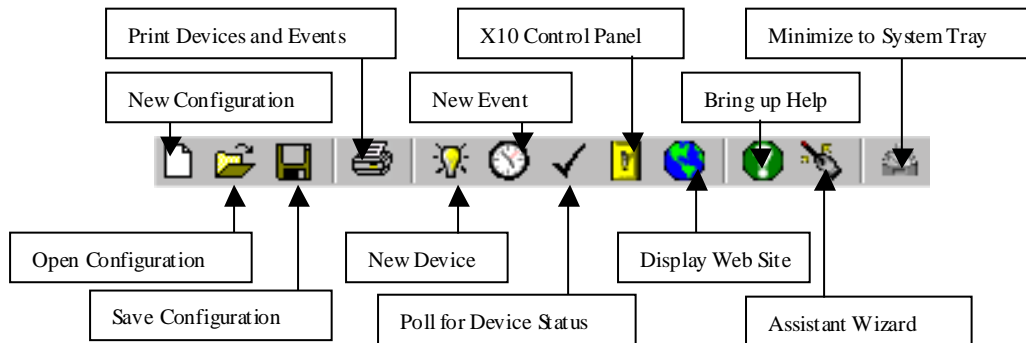
Checking this box will allow all events to be logged into memory to be visible in the event log view. If you are running a slower system, you may want to disable this feature to increase X10 response time. You can still log all events to the log file.

## 1.17 Max size for Event Log (memory only)

This option is available from the menu *View->Options->General*.

All event activity since HomeSeer was started is stored in memory. This can start to use up a lot of memory over time. Enter the maximum amount of memory you would like to reserve for the event log. When this memory is full, HomeSeer will start deleting the oldest events, while adding new events to the log. The log will not use more memory than you have allocated to it. The default size is 100,000 bytes, and maximum size you can enter is 200,000,000. Each log entry uses about 60 bytes.

## 1.18 Toolbar



The following buttons are available via the tool bar. The toolbar may be hidden by unchecking **Toolbar** in the **View** menu.

### Create New Configuration

Creates a new empty configuration. It will not have any devices or events.

### Open a Different Configuration

Prompts you for a different configuration to load. You may want to create a separate configuration for when you are away on vacation.

### Save Configuration

Prompts you for a filename to save your current configuration. This is useful if you want to create a new configuration that is slightly different from your standard one. It can also be used to back up your current configuration. Note that this will change your current configuration and make the one you just saved the active one. HomeSeer will load this new configuration next time it is run. There is no need to save any changes you make to your configuration. HomeSeer automatically saves all changes as you make them.

### Print

Prints out a listing of your devices, your events, or the event log, depending on the current view.

### Add Device

Pops up a dialog allowing you to add information about a new device. The device is then added to the device list.

### Add Event

If you are in the Events View, a new unnamed event will be added to events list. Its trigger will be set to manual. It will have no global actions and will not control any devices. If you are in the Devices View, a new event will be created that will control the last device selected. It will have a trigger set to manual and will set the device action to ON.

### Get Device Status

This will poll all X10 devices that support the X10 *Status Request* command. It will then update the device list with the new status. Note that very few devices support this feature. Some that do are the LM14A and PLM21A lamp modules.

### Launch X10 Control Panel

This will bring up a dialog that lets you control any X10 device directly. See the X10 Control Panel section for more information.

**View web page**

Launches your web browser and displays a web page containing all your devices and events. This is the same page you connect to remotely from the Internet.

**Help**

Brings up the HomeSeer help file.

**Minimize to System Tray**

Minimizes the application to the system tray. Right clicking the HomeSeer icon in system tray will pop up a menu allowing you to restore or exit the application. You can also set an option in *View->options->general* so that HomeSeer always minimizes to the system tray.

## 1.19 Controlling your devices via email

You can control HomeSeer via email. From an email message you can do the following:

- Send an X10 command by housecode and devicecode
- Send an X10 command by device name
- Force the trigger of an event

The command must reside in the subject line of the email message. To setup HomeSeer to receive email commands do the following:

- Create a new event and set its trigger to email reception.
- Set the EMAIL *from* field to the address that is authorized to send you email commands. You can click the all addresses box, but this will allow anyone to control your devices via email. You most likely do not want this type of open access.
- You do not have to set any actions for this event.

The format of the email is as follows. Remember to enter this information in the emails subject line. Note that none of the characters on the subject line are case sensitive. The command "EXECX10" is the same as "ExecX10".

Controlling devices directly with Housecode Unitcode

```
execX10:devices,dimval
```

To send an off command to devices A1 and A2, the command would be:

```
execX10:A1+2,OFF
```

To dim device B2 50%:

```
execX10:B2,DIM,50
```

**Controlling a Device by Name**

You can send an X10 command to a device using the device name. Assuming that there is a device named "table light" that is located in the "den" and you want to turn it on, the command would be:

```
execX10ByName:den table light,ON
```

**Forcing an Event to trigger**

You can include the name of an existing event in your email. If you would like to trigger the event named "all lights off", put the following command in the subject line:

```
triggerevent:all lights off
```



# Chapter 2

## 2 Working with Devices

You must tell HomeSeer some information about your X10 devices. Adding devices has to be done before any events are created. Add as many X10 devices as you would like. Bring up the *Add Device* dialog either from the toolbar or by right clicking in the devices pane.

This same dialog may be used to edit any existing device. Simply right click the device and select *Device Properties*.

### 2.1 Supported Devices

HomeSeer supports the following types of X10 devices:

- Any device that supports the standard X10 Dim and Bright commands
- Any device that supports the X10 ON and OFF commands
- Any device that supports the X10 Status Request command. These devices can be polled for their ON/OFF status
- Any device that supports the X10 pre-set dim command. These devices can be dimmed to a specific dim level with going to full bright first. This includes *PCS* and *SmartLinc* switches.
- Leviton wall switches such as the 6381. These devices use the X10 Extended Code command. The Extended Code command sends the command plus two bytes of data. HomeSeer sends the first byte as the dim level (between 1-63), and the second byte as 30(hex). This second byte is the direct dim command. These switches support Scenes also. HomeSeer does not directly support the scenes function, but you can create a script to send these functions. See the *Scripting* chapter for more information.

**Note that some X10 controllers cannot receive the Extended X10 command. If your X10 controller uses the TW523 interface, then it will not see any Extended X10 commands. This means that HomeSeer will not be able to track the dim level of any light that is controlled externally to HomeSeer. If HomeSeer sets the dim level, then it will be represented properly in the devices view.**

- SmartLinc SwtichLinc wall switches including programming the default ON level and ramp up speed. This also includes the SmartHome LampLinc modules.
- Lightolier Compose switches that use the *Compose* protocol.
- RCS TX15B, RCSTR15 thermostats and compatibles as well as the HAI RC80 and Enerzone.
- Stanley garage door sensor (must use the supplied script to support this. See the script "sense\_garage\_door.txt")

### 2.2 Creating/Editing devices

Add as many X10 devices as you like. Bring up the Add Device dialog either from the toolbar or by right clicking in the devices pane.

**Figure 3 Create/Edit Device Dialog**

This dialog may also be used to edit any existing device. Simply right click the device and select Device Properties.

You can delete a device by right clicking the device and selecting Delete Device.

Use the following steps to create a new X10 device:

1. At the main window, click on "Devices" in the "views" pane. A list of all previously added devices will be displayed in the top pane.
2. Either click the light bulb on the toolbar, or use the mouse and right click in the "Devices" pane to bring up a pop-up menu. Select Device Properties... from the menu. A new dialog will appear.
3. Enter a descriptive name for the device, such as "table lamp"
4. Enter the location of the device. This is typically the room where the device is. The location is a drop down box that lets you select from any other locations you may have already added. Either select an existing location or type in a new one. Note that it is much easier to find your devices if you group them by location. You can then sort the device views by location later.
5. Select a device type for this device. Use the drop down box to choose an existing device type. Note that any device type selected here will set the correct check boxes: "Device can be dimmed", "Dim Type", and "Device supports status response". If you do not see a device type that matches your device, select the closest match. You can then check (or uncheck) the appropriate check boxes. Note that the check boxes are what dictate the attributes for the device and they will override the settings for the selected device type. It's best to use the device type



selection. You can create your own device types from the Options dialog. You can bring up the Device Types dialog from View->Options->Device Types. HomeSeer supports the following types of devices:

- Any device that supports the standard X10 Dim and Bright commands
  - Any device that supports the X10 ON and OFF commands
  - Any device that supports the X10 Status Request command. These devices can be polled for their ON/OFF status
  - Any device that supports the X10 pre-set dim command. These devices can be dimmed to a specific dim level without going to full bright first.
  - Leviton wall switches such as the 6381. These devices use the X10 Extended Code command. The Extended Code command sends the command plus 2 bytes of data. HomeSeer sends the first byte as the dim level (between 1-63), and the second byte as 30(hex). This second byte is the direct dim command. These switches support Scenes also. HomeSeer does not directly support the scenes function, but you could create a script to send these functions. See the Scripting section for more information.
  - SmartLinc SwitchLinc or Compose wall switches or LampLinc modules, including programming the default ON level and ramp up rate.
6. Enter the X10 code for the device. This is the code that you set with the small knobs on your X10 modules. It's best to use a different house code for each room. HomeSeer puts no restriction on what house and device codes you use. Use "F" for the family room, "O" for outside lights, "D" for the Den, etc. X10 devices use house codes "A" through "P". You can enter housecodes "q" through "z", and unit codes 17 through 64 and use these housecodes and devicecodes as "virtual" devices. A virtual device is not an actual X10 device, but a device that can hold status. You may want to keep status on whether someone is home or not. This is a good way to keep track of this.
  7. If this is a SwitchLinc, Compose, or LampLinc device, you can program the default ON brightness level and the ramp up brightness rate by pressing the "options" button.
  8. The *Hide Device from Views* checkbox is useful if you do not want the device to appear on your web site for security reasons. If this is checked, people who log into your web site as guest will not see this device. You can show/hide the device from the Windows interface by selecting *Show All Devices* from the *View* menu.
  9. The *Include in power fail recovery* tells HomeSeer to set this device to it's proper state when recovering from a power failure. The power failure recovery feature must be enabled in the options.
  10. Check the *Voice Command* option if you would a voice command created to control this device. Check the *Confirm* box if you wish HomeSeer to ask if you really want to perform the command. A voice command will be created in the following format ([ ] = word is optional):

[turn] [on] [off] [the] <device location and name> [on] [off]  
 or  
 [set] [dim] [bright] [the] <device location name> [by] [to] <number> [percent]

If the device location name is *den floor lamp*, then you can say any of the following phrases:

*turn off the den floor lamp*  
*den floor lamp off*  
*set the den floor lamp off*  
*turn on den floor lamp*  
*dim the den floor lamp by twenty five percent*  
*set the den floor lamp to fifty percent*

*den floor lamp one hundred*

## 2.3 Creating/Editing an X10 device

Only change the following if your device type was not available in the "device types" drop down list. The preferred way to create a new device type is through the device types tab in the options. If you create a new device type there, the new device type will be available when creating new devices.

To create a device that uses the *Lightolier Compose* protocol, use the device types tab in the options.

1. Select the appropriate dim type. If your device supports the "Preset Dim" X10 command, you can select it here. If your device is a Leviton 6381 wall switch, then select "Leviton Dim".
2. If your switch is a SmartLinc SwitchLinc wall switch, select the "SwitchLinc" checkbox. You will also have to select "preset dim" and "Supports Status Request". You can program some features of the switch such as the default ON level and ramp up brightness rate by clicking the "options" button.
3. Enter the type of device you are controlling. Any X10 module or wall switch that allows dimming should have "Light" selected. If the module controls an appliance, select "Appliance Module". This tells HomeSeer not to allow the user to send "Dim" or "Bright" commands to this module.
4. Select the "Device supports status response" if the device can be polled for its "ON" "OFF" status. Many modules do not support this ability. If you are unsure if your module does, press the "Check" button, and HomeSeer will check the device for you. If it gets a response, it will put a check in the appropriate checkbox.
5. Select "Status Only" if you are using this device to display some status information. If this is checked, no control buttons will appear on the web page, and you will not be allowed to control the device using its context menu. This feature is useful if you are setting the status of the device using a script. See the scripting function *SetDeviceString*.

## 2.4 More on Virtual Devices

Virtual devices are devices that act like X10 devices, but do not have the ability to actually receive or send X10 commands. You can create a virtual device by selecting the lower case house codes in the range "q" through "z", and you can use unit codes 17 through 64 on all house codes. You set the status of these devices just as you would for real X10 devices. When you set the X10 actions for an event, you will notice that your virtual devices are listed with all your *real* devices. When an event triggers, you can set the status of one of your virtual devices to "ON", "OFF", or "DIM" to a specific brightness. An example use for a virtual device might be creating a flag that tells HomeSeer that someone is home or not. Depending on the flag, your events may take different actions. Here is an example:

1. Create a virtual device "q1" and call it "I am Home"
2. Create a new event and call it "I have arrived"
3. Set the trigger for this event to "X10 Command A1 ON"
4. For the actions for this event, set the device "q1 ON"
5. Create another event named "I am leaving", set the trigger to "A1 OFF" and the action to "q1 OFF"
6. When you are home, touch a keypad so it sends the X10 command "A1 ON", and when you are away, touch "A1 OFF" (this will set the state of the virtual device "q1" to either ON or OFF)

Now you have a "flag", that indicates when you are home. No other X10 action will modify the flag. You can now use the flag with the conditional trigger. This allows you to control your lights differently when

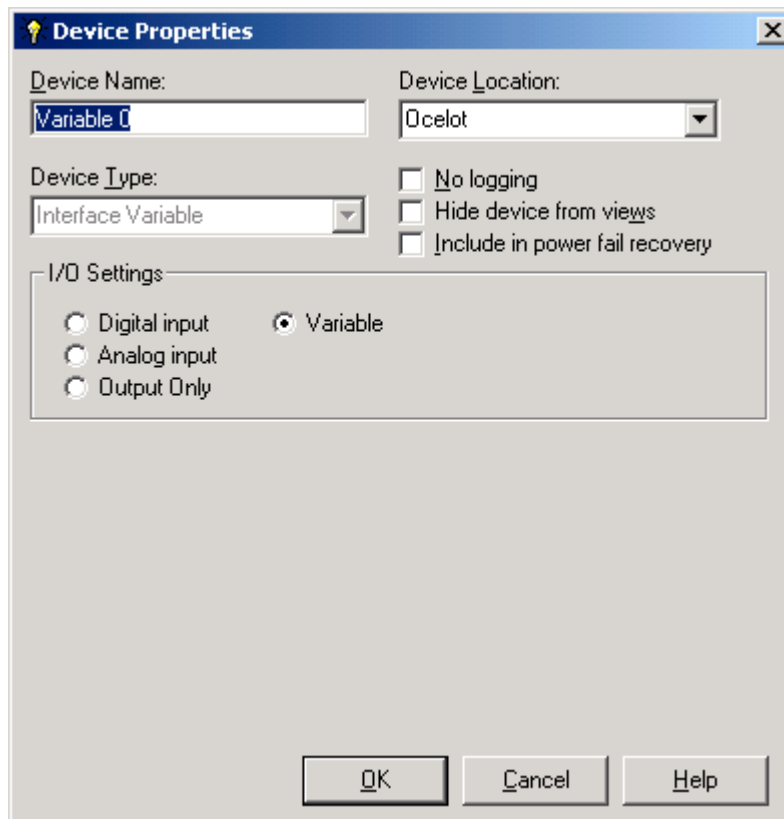
you are home as opposed to when you are away. To have a light go ON at 7:00pm when you are NOT home, create a condition like this: (conditions are created in the event properties, trigger type "conditions")

```
IF DEVICE "I am Home" IS OFF
AND
TIME IS 7:00PM then
light ON
```

Note that the status for the virtual devices (as well as all devices) is also available to any scripts you write. So you could create even more sophisticated logic by using the scripting language.

## 2.5 Devices that Represent Input/Output Points and Variables

With some home automation controllers, it's possible to have HomeSeer display the controller's input/output points, and system variables. For example, on the *Applied Digital Ocelot* HomeSeer can display all 128 system variables as devices as well as input/output points on the SECU16 module. These variables appear in the HomeSeer device list. You can then turn variables *on* and *off* (sets their value to 1 and 0), and also set the variables to absolute values. To create the devices, click on the setup... button on the interfaces tab in the HomeSeer options. After creating the devices, the device's properties will have different properties like this:



**Figure 4 Device Properties for Variables and I/O**

From the properties, you can change the input/output type, or change the device to a variable. Note that the proper type should be set by default. You may have to change the input/output property if HomeSeer cannot determine the type itself.

To change the value of a variable, use the device action tab in the properties of an event. The device action will allow the setting of the devices value. The value will be saved to the variable when the event

is triggered. You can also use normal On/Off commands with a variable. This will set the variable to a 1 or a 0 respectively. With some controllers, a drop down list will be presented that allows you to select predefined variables.

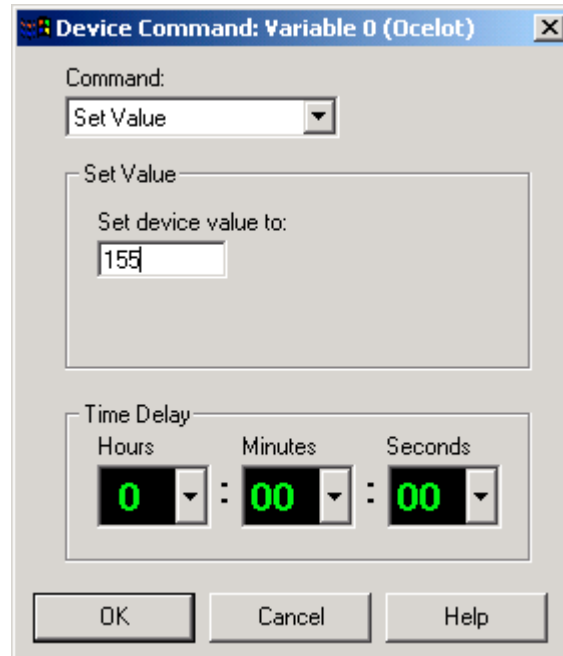


Figure 5 Setting a Controller Variable

## 2.6 Changing the status display

Normally, the device status display screen displays the X10 status of your device. If you like, you can display a custom text string in place of the normal ON,OFF,DIM, etc. display. On the device action tab in the event properties, simply enter the new text you wish displayed for the device. For example, if you would like the device status to be "Door Open" for a device that detects whether a door is open or closed, simply enter "Door Open" in the text box. To clear any text displayed and restore the default display, enter *clear* in the text box.

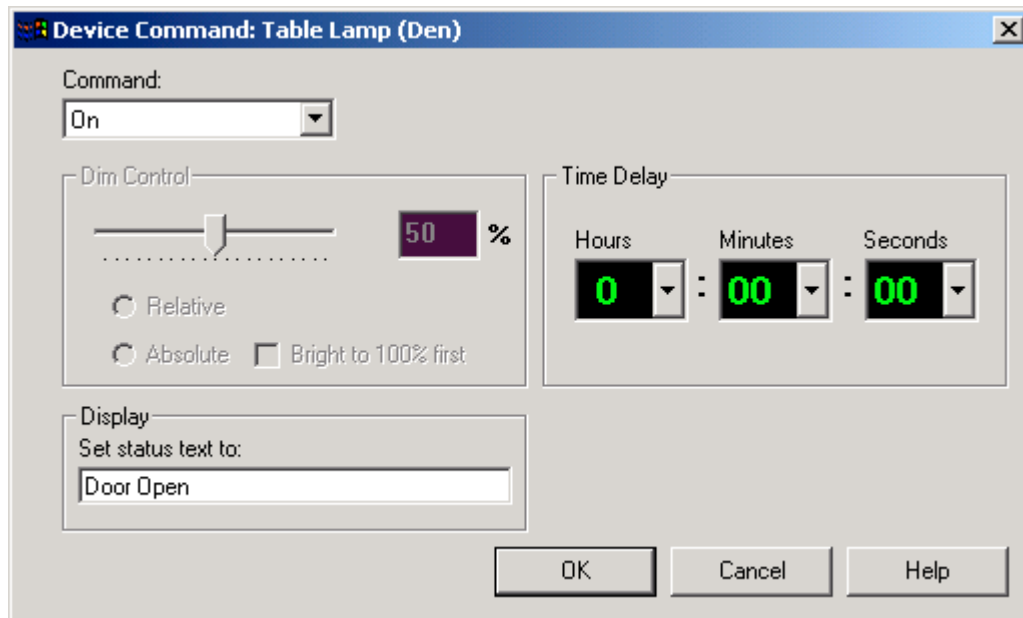


Figure 6 Setting custom device status

## 2.7 Creating Custom Device Types

You can create your own device types in *View->Options->Device Types*. Enter a name for the device type, such as "Lamp Module" and select the attributes of the device with the check boxes. Click the *Add/Update* button to add/update your device type to the list. This new device type will now be available when you create new devices.

**Note: When changing any attributes of a device type, the changes will not be reflected in any devices that are set to this device type until the device is edited and the device type is re-selected.**

If your device is a thermostat, select the *Thermostat* checkbox, and then select the proper script for your thermostat. Scripts to control the RCS TX15B, RCSTR15, and RC80 are available. Scripts for other thermostats will be made available from the HomeSeer website support page.

Select the *Compose* check box if the device supports the *Lightolier Compose* protocol.

## 2.8 Deleting Devices

To delete a device, right click the device and select *Delete...* The device will be deleted from the device list. You may select and delete multiple devices by holding down the *Control* key while selecting.

**Be careful when deleting devices. If the device is being controlled from an event, it will be automatically removed from the event. You can check to see if a device is being controlled by any of your events by selecting it in the *device view*. Any events that control this device will appear in the lower event window.**



---

# Chapter 3

---

## 3 Working with Events

### 3.1 Events Overview

There are two types of events; events that control a single device and events that control multiple devices. Events that control multiple devices are referred to as *macros* by other applications. You can create events in either the *Device View* or *Events View* screens. When you create an event in the *Device View* screen, the event controls a single device. When you create an event in the *Event View* screen, you have the option of selecting multiple devices.

### 3.2 Creating an event for a single device

1. Click the *Devices* icon in the View pane
2. Make sure the events pane is displayed, right click any device, then select *Show Events*
3. Click on the device you wish to create an event for
4. Any events currently defined for this device will be displayed in the lower pane
5. Click the *Add Event* button, or right click in the events pane and select *Add Event*.

*Note: you must select a device before you press the Add Event button.*

6. A new event will appear in the events pane. The name of the event will be the same as the name of the device it controls.
7. Right click this new event to edit its trigger and global actions. The default trigger will be set to manual. This means the event can only be triggered by selecting *Execute Now* after right clicking the event name

### 3.3 Creating an event that controls multiple devices

1. Click on *Events* in the View pane.
2. Click the *Add Event* button, or right click on the event list pane and select *Add Event*.
3. The *event properties* dialog will appear and the device name will be set to *UnNamed*. Enter a new name for the event. Note that quote (") characters are not allowed in event names.
4. Select the *X10 Device Actions* tab. Select the device you wish to control from the upper pane, and drag it to the lower pane. The device will be added to the list of controlled devices in the lower pane. If you wish to control a device that is not in the list, click the *Add Device...* button and you will be prompted for information on your new device. The device will then be added to your device list. To modify the command of an existing device, double click (or right click) the device. You will be prompted for its command. See the *Event Actions* section for more information. You may add the same device more than once if you wish to send it multiple commands, or set a different offset delay for each one. This is useful for ramping up

the light level of a lamp over time for example.

5. If you wish, you can set a delay for the X10 action of the selected device. In the X10 actions dialog, set the action for the device such as ON, then select a time delay offset. The offset is in the range of 1 second to 24 hours. Each device in the event may have a different delay time.
6. Select the *Trigger* tab to set a trigger for this event. All other tabs are used to set actions for your event.



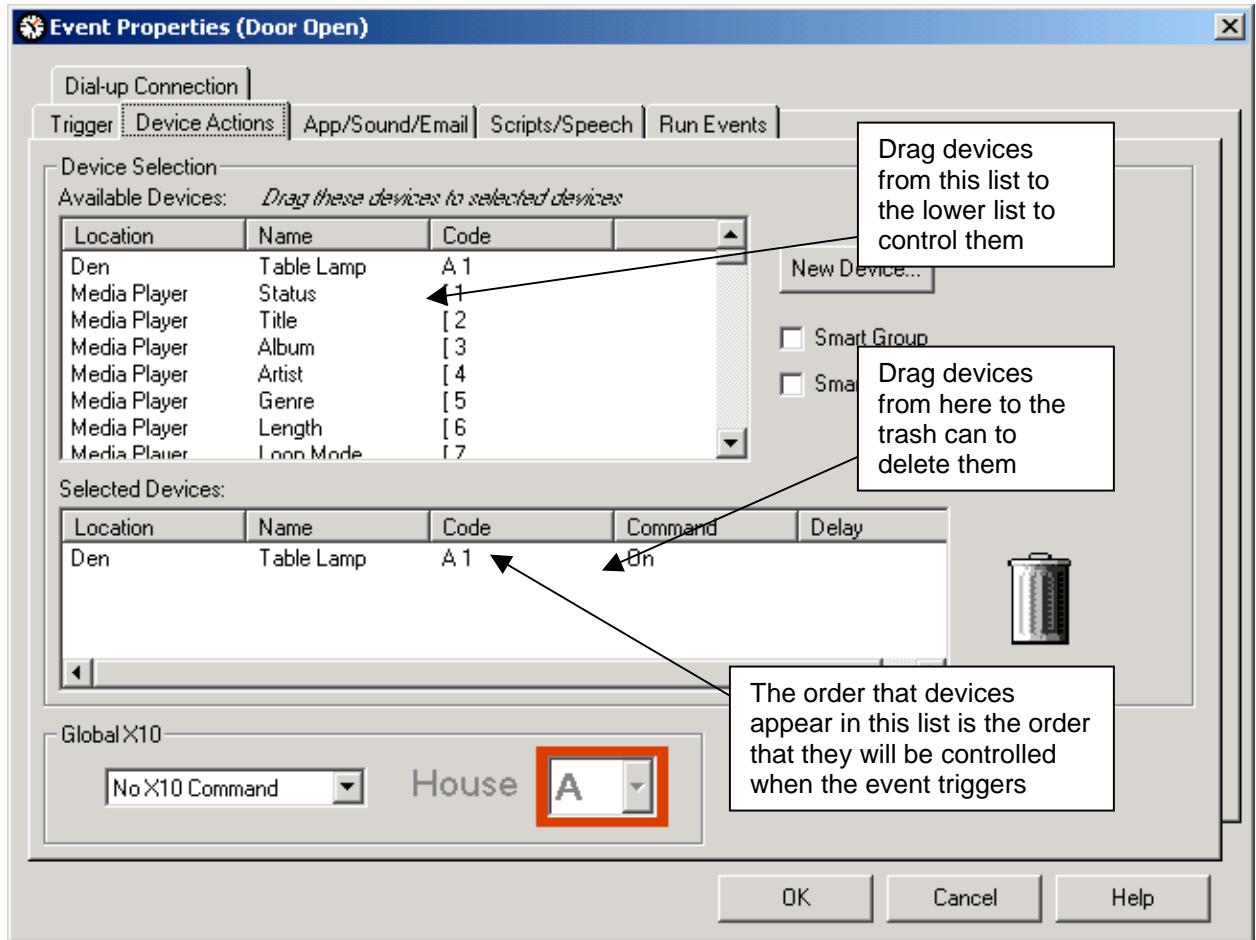


Figure 7 Device Selection Dialog

### 3.4 Deleting Events

In the *Events View*, right click the event you wish to delete, then select *Delete Event...*. You will be asked if you really want to delete the selected event. You may hold down the *Control* key on the keyboard to select multiple events.

### 3.5 Enabling an Event as a Voice Command

If you would like to be able to speak commands, you can assign a voice command to each event. To create a voice command, do the following:

1. Click on a event in the *Event View*
2. Right click the event, then select *Event Properties*
3. In the *Voice Command* text box, enter the voice command you would like to speak to trigger the event
4. From the *Enable* drop down list, select the input device that will be used to give the command. Normally this is *Mic* (your computers microphone). If HomeSeer Phone is installed, you can also select *Phone* or *Both*. This allows you to create commands that only work from the microphone or only over the telephone.

**Note that if you disable an event, the voice command will be disabled also.**

### 3.6 Event Triggers

A trigger is used to tell the event to perform some action. You can set the trigger to *manual*, which forces the event to be triggered by either the execute event menu, by a *voice command*, or by another event. Manual events are useful as macros. They define a set of actions that may be triggered by some other event. There are many types of triggers in HomeSeer:

- An absolute time and/or date
- At Sunrise
- At Sunset
- By X10 command
- By condition
- By email received
- At a specific interval of time (recurring)
- By a voice command
- By an infrared match
- By a change in status of a device
- By a change in value of a device
- Security panel event
- Forced to execute from the web page
- Manually
- From a plug-in

Edit the trigger by right clicking an event and selecting *Event Properties...*

Select the type of trigger you want from the drop down combo box labeled *Type*. The current trigger settings are displayed by default.

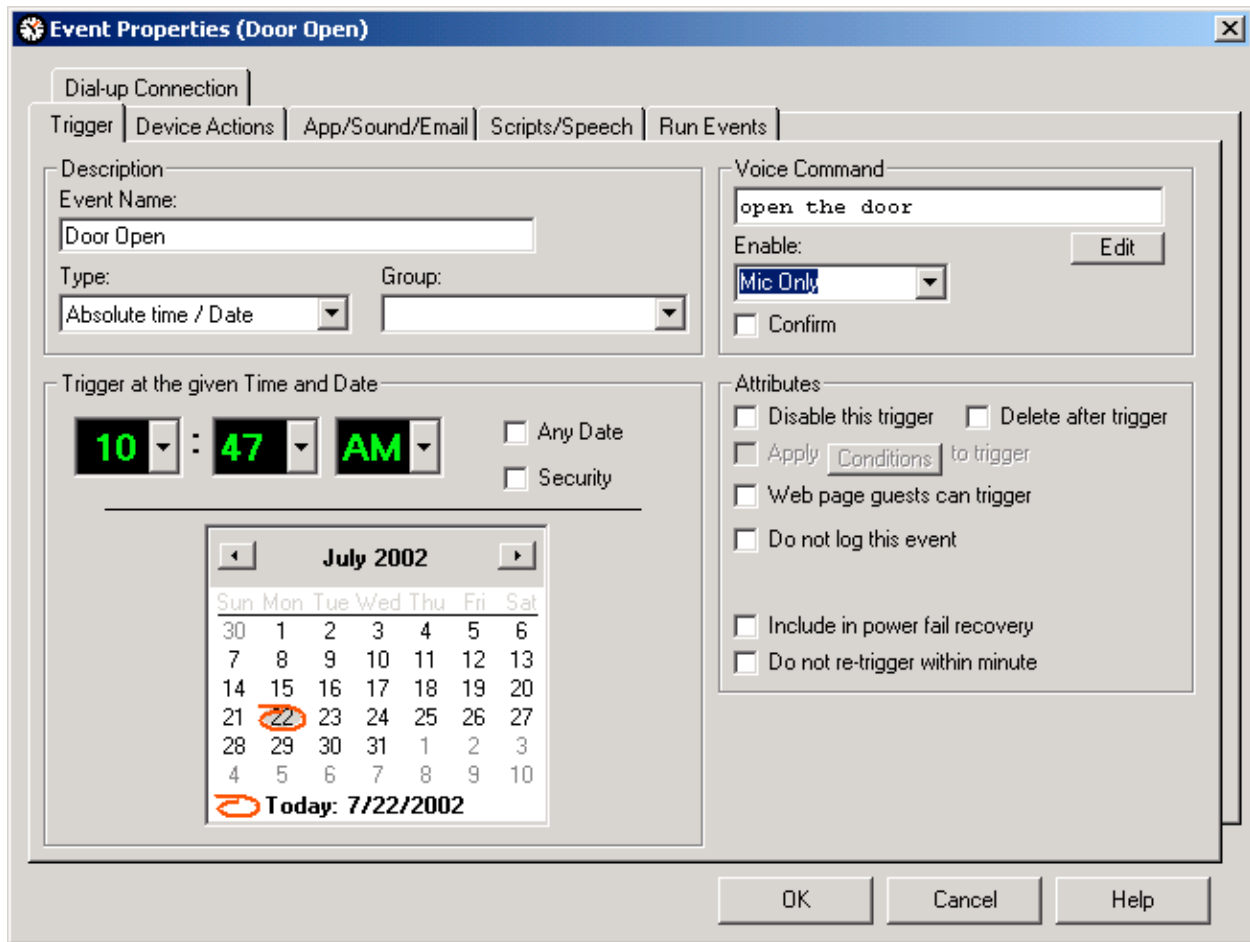


Figure 8 Event Properties Dialog

The following options are available on the Event Properties dialog and pertain to all triggers.

#### Event Name

Enter the *name* you wish to call this event. Although you can name multiple events with the same name, this can cause confusion. Events can be triggered by name. If you would like to send an email message that forced an event to trigger, you would use its name. If you have multiple events with the same name, HomeSeer will only trigger the first one. Try to keep all event names unique.

#### Type

This drop down list selects the event type. Depending on the event type selected, different options will appear in the dialog

#### Group

Select a group name from the drop down list, or type in a new group name. Groups are useful for organizing events so that they may be viewed easier in the event list.

#### Voice Command

Enter a command string that you would speak to trigger the event. Check the box labeled *Voice Command* to enable the command. Note that you must have voice recognition installed to use voice commands. You can enable voice recognition on the *Voice Recognition* tab in the options. See the Voice Recognition section for more information on formatting voice commands.

**Enable**

The Enable drop down list allows you to select the input the device that will accept the voice command. The options are:

- **Mic** (use the microphone attached to your computer)
- **Phone** (Enable the voice command when a user is interacting with the computer over the phone. Only available if HomeSeer Phone is installed)
- **Both** (Allow the voice command to work over the microphone and the telephone)

**Confirm**

If checked, the user must confirm the voice command before the events actions are executed.

**Disable this Trigger**

A trigger may be disabled without deleting it. Check the box labeled *Disable this Trigger* and the event will not trigger. You can also disable a trigger by right clicking it in the event view and selecting *Disable/Enable Event*. This will toggle its enabled status.

**If you would like to disable all of your events, select *Disable All Events* from the tools menu.**

**Delete Event after Trigger**

If you would like the event to only run once, check the box *Delete Event after Trigger*. This will permanently remove the event only after it has been triggered. This is useful for creating one-time events like recording a channel on a VCR.

**Apply conditions to trigger**

Some trigger types allow you to apply conditions to them. If the trigger allows this feature, this option will be enabled. The *recurring* trigger allows this for example. You can setup a recurring event to run every hour, and apply some conditions to only allow it to trigger during the day. Select the trigger type you want, then click the *conditions* button to setup conditions for it.

**Web Page Guests can Trigger**

When guests log into your HomeSeer web site they can view the status of your devices and see your events. They cannot make modifications to your device configurations, events, or force a trigger of an event or control a device. There may be situations where you might want to allow a guest to trigger some action for a subset of your devices. If you enable this checkbox, guests will be able to trigger this event. This can be used for moving a web cam for example, or allowing the control of a particular lamp to get someone's attention.

**Note that the *Web Page Guests can Trigger* option affects the triggering of events only. If you would like to allow a guest to control a particular device, assign the device to an event and give the guest access to that event.**

**Do not log this event**

When selected, the event will not write any information to the event log when it triggers. This is useful for events that are triggered by a motion sensor and tend to clutter the log with motion-sensed messages.

**Security**

Some triggers can have their trigger time offset using Security Mode. See Security Mode for more information.

**Do no re-trigger within minute**

This option is available on some triggers. It will keep the event from triggering for one minute. This is useful for events controlled by motion detectors, and can reduce double triggers from keypads.

**Include in power fail recovery**

If power fail recovery is enabled, this event will be run when HomeSeer detects that it is recovering from a power failure. If the event is not controlling a critical device, maybe it simply speaks the time or performs some other non-critical function, then this checkbox should be left unchecked. If the event controls a critical device, maybe it turns on the outside lights at sunset, then this checkbox should be checked. If checked, HomeSeer can then be sure that the status of the controlled devices is correct. See the power fail recovery chapter for more information.

### 3.7 Manual Event Trigger

You can set an event to not have a trigger by selecting a trigger type of *manual*. These types of events could be referred to as *macros* since they are not triggered automatically. They can be triggered from other events however, or from the *Run* button on the *Events* web page.

### 3.8 Recurring Event Trigger

An event can be set to trigger at regular intervals. You can have a script run every minute, or turn off a device every 5 minutes. When setting the trigger to *recurring*, you can check the checkbox *reference to hour*. This will cause the interval to be an offset from the hour. This can be used to set a recurring trigger to happen every 15 minutes, and it will trigger 15 minutes past the hour, 30 minutes past the hour, 45 minutes past the hour and on the hour. If the interval was set to 5 minutes, it will trigger at 5 minutes past, 10 minutes past, etc. *Reference to hour* is ignored if the interval is over 60 minutes.

Note that the event will always trigger on the hour.

If you want the event to trigger *only* on the hour, set the interval to 60.

The screenshot shows a configuration window for a recurring trigger. At the top, there are two dropdown menus: 'Type' set to 'Recurring' and 'Group' set to 'system'. Below these is a section titled 'Trigger continuously at the given interval'. Inside this section, there are two input fields: 'Run every 15 Minutes' and '0 Seconds'. Below the input fields, there is a checked checkbox labeled 'Reference to hour (60=On the hour)'. Underneath this checkbox is another unchecked checkbox labeled 'Only trigger once after the hour'.

Figure 9 RecurringTrigger

### 3.9 Trigger Event by Absolute Time

This will trigger the event at an *absolute time* or *date*. You can also specify the day.

1. Use the drop down lists to specify a time.

2. Select the day or days you wish this event to trigger.

If you would like the event to trigger only on a specific day, uncheck the box labeled *Any Date*. A calendar will appear that will allow you to select the actual day you would like the event to trigger. Note that the *Days* setting is ignored when a date is selected.

The screenshot shows the HomeSeer event configuration interface. At the top, there are two dropdown menus: "Type:" set to "Absolute time / Date" and "Group:". Below these is a section titled "Trigger at the given Time and Date". It features three dropdown menus for time selection: "1" for hours, "41" for minutes, and "PM" for the period. To the right of these are two checkboxes: "Any Date" (unchecked) and "Security" (unchecked). Below the time selection is a calendar for July 2002. The calendar shows days from 30 to 10. The date 22 is circled in red, and a red arrow points to it. Below the calendar, it says "Today: 7/22/2002".

Figure 10 Trigger by Absolute Time

### 3.10 Trigger Event by X10 Command

HomeSeer listens to all X10 commands that are sent over your household wiring. You can set events to trigger when a particular command is seen.

For example, in your family room you could have an X10 table top controller set to house code A. The first button would control device A1. When you pressed this button, the command A1 is then sent over your wiring and received by HomeSeer. Instead of setting a device to code A1, you could set an event to *trigger* on this code. The actions you can then perform are limitless. You could have HomeSeer dim all the lights in the family room in preparation for a movie, turn off all the lights in the house, send email, play a sound, launch an application on your computer, send I/R commands to your A/V equipment, or whatever you want. You set triggers on the device code and the X10 command. This allows you to set triggers on *X10 ON* and *X10 OFF* commands for the same device.

The dialog has the following options:

#### Address

Enter the X10 house code and unit code you wish to trigger this event. Note that you can set the unit code to *Any* and any unit code will trigger the event as long as the housecode matches.

#### Command

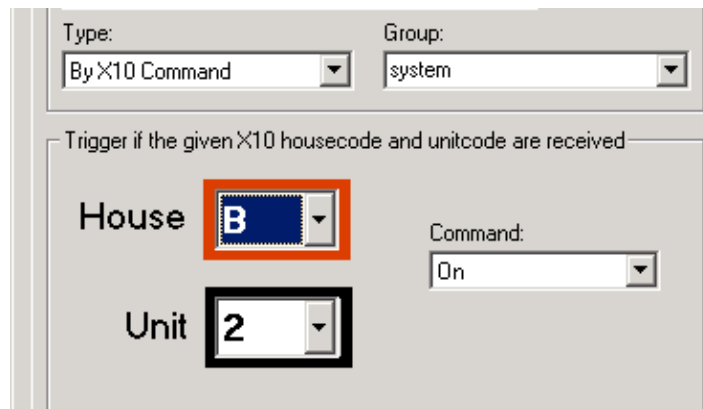
Enter the command associated with the device. You would normally select either ON or OFF , however, all 16 X10 commands are available. You cannot use the *reserved* entry. You may select *Any* to trigger when any X10 command is received.

#### Days

You may select the days this trigger is valid.

#### Apply Conditions to Trigger

You can apply some conditions to this trigger by checking this checkbox. Press the Conditions button to set conditions for the trigger.



The screenshot shows a configuration dialog for a trigger event. At the top, there are two dropdown menus: 'Type' set to 'By X10 Command' and 'Group' set to 'system'. Below these is a section titled 'Trigger if the given X10 housecode and unitcode are received'. This section contains three dropdown menus: 'House' set to 'B', 'Unit' set to '2', and 'Command' set to 'On'. The 'House' and 'Unit' dropdowns are highlighted with red and black boxes respectively.

Figure 11 Trigger by X10 Command

### 3.11 Trigger Event by Status Change

Events may be triggered when the status of a device changes. The event will trigger no matter how the status of the device changed. If HomeSeer sends an ON X10 command to a device, and the device is OFF, the event will trigger. This works on X10 devices, and virtual devices.

**Figure 12 Device Status Change Dialog**

Note that the device must actually change its status before the event will trigger. The types of status changes are:

- Device goes to the ON state
- Device goes to the OFF state
- Device goes to the Dim state
- Device goes to Any state

### 3.12 Trigger Event by Sunrise/Sunset

The event will trigger at either *Sunrise* or *Sunset*. You must tell HomeSeer where you live by entering your longitude and latitude under the menu *View->Options->Sunrise/Sunset*. Here you can either pick a city close to you or enter your exact longitude and latitude. HomeSeer automatically adjusts the Sunrise/Sunset times at midnight every night.

The *trigger* dialog will display the current Sunrise and Sunset times. Note that daylight savings is handled by your computer. Make sure your computer is set to handle daylight savings or your sunrise and sunset times will be incorrect. This is usually set by double clicking the clock in your Windows system tray.

#### **Sunrise/Sunset offset**

If you wish the event to be triggered at a specific offset from the actual sunrise or sunset time, you can enter it here. For example, if it is still somewhat light outside when actual sunset occurs, you may want to delay turning on your outside lights for 30 minutes. Set the event to trigger to 30 minutes after sunset.

You can set the offset up to 12 hours before or 12 hours after sunrise/sunset.

#### **Days**

Select the days you wish to be included in this trigger.



The screenshot shows a configuration window for an event. At the top, there are two dropdown menus: 'Type' set to 'At Sunset' and 'Group' set to 'system'. Below this is a section titled 'Trigger at sunrise/sunset with the given offset'. It contains two digital display boxes showing '0' and '00' with dropdown arrows. To the right of these boxes are two radio buttons: 'Before' (unselected) and 'After' (selected). Further right, the times 'Sunrise: 5:26 AM' and 'Sunset: 8:21 PM' are displayed. At the bottom left of this section is a checkbox labeled 'Security' which is currently unchecked.

Figure 13 Trigger by Sunrise/Sunset

### 3.13 Trigger Event by condition

Conditions give more flexibility in determining when an event should trigger. You can set as many conditions as you want on a trigger. The event will trigger only when all of the conditions are true. If you need more flexibility than the conditions allow, you may want to consider writing a script to trigger your event. See the scripting chapter for more information on scripting. Here are the types of conditions available:

#### Time conditions

- An absolute time value
- Before a specific time
- After a specific time
- Anytime before sunset
- Anytime after sunset
- Anytime before sunrise
- Anytime after sunrise
- Minutes before sunset
- Minutes after sunset
- Minutes before sunrise
- Minutes after sunrise
- Day time (after sunrise but before sunset)
- Night time (after sunset but before sunrise)

#### Device conditions

This is used to set a condition on the status of an X10 device

- Whether the device is ON
- Whether the device is OFF
- Whether the device has been ON for a specified amount of time.
- Whether the device has been OFF for a specified amount of time
- Whether a device has just changed its status (supports On, Off, Dim, Bright, or ANY)
- The device has been ON for at least a particular amount of time (keeps triggering after specified time as elapsed)
- The device has been OFF for at least a particular amount of time (keeps triggering after specified time as elapsed)

### 3.14 How Event Conditions Work

Conditions are checked once every minute and when an event occurs (like an X10 signal received or device state change for example). All conditions are checked only once and an action for a condition cannot affect any other conditions during the current check. For example, suppose we have the following two conditions. These conditions would allow a wall switch to turn a light on if it's off, or brighten the light by 10% if it's already on.

- If received A1-ON AND B1 is OFF then turn B1 ON
- If received A1-ON AND B1 is ON then brighten B1 10%

Let's assume B1 is OFF. HomeSeer receives an A1-ON. HomeSeer does the following:

- Condition 1 is now true (received A1-ON and B1 is OFF) B1 goes ON
- HomeSeer does not update the status for B1 until ALL conditions have been tested.
- Condition 2 is NOT true, and does not execute (HomeSeer still thinks B1 is OFF)
- After all conditions are tested, the status for A1 is set to ON and the status for B1 is set to ON.

**Why, if the status of a device is changed in the event, is the change not detected during that event?**

**If the status for the device was updated immediately, both conditions would become true and the light at B1 would go on, and then brighten 10%. (This is probably not what you want.)**

**Status of devices changed by a condition are not updated until all the conditions are checked.**

**All conditions must be true for the event to trigger.**

**X10 commands received will reset the timer associated with the device. If device A1 was on for 10 minutes and an A1-ON was received, then the timer for A1 is reset back to 0.**

**If you enable the SmartOn feature for a device, then X10 commands sent by HomeSeer will not reset the ON time for a device if it is already ON.**

**You can use the Status Change trigger to trigger events when the status of a device actually changes. Status changes trigger events immediately.**

Conditions take the form:

*IF (condition) AND (condition) AND ....*

*OR*

*IF (condition) AND (condition) AND ...*

*THEN (take some action)*

For example, suppose you would like a light to be turned off if it was left on for 30 minutes. Assuming the light was named closet light:

*IF closet light has been ON for 30 minutes then turn it off*

In the trigger dialog, set the trigger to conditions, select the closet light device from the first drop down list box, then select: *Has been on for* in the second drop down list box. Set the time for :30. You would then add the *closet light* device to the events device list and set its action to OFF. This will turn the light off after it has been on for 30 minutes. This only works if you turned the light on from a switch that sends X10 commands over your house wiring so that HomeSeer knows when it was turned on. If an event turns a device ON or OFF, HomeSeer knows the status.

Here is another example. Suppose you would like all the lights downstairs to go off after you go to bed. You would use a keypad to tell HomeSeer that you are going to bed. HomeSeer treats all devices the same; that is, even a switch has an ON/OFF status. You would set it up like this:

1. Create a device that is nothing more than the switch that will send the bedtime command. Label it Switch A1 ON.
2. Create a new event and label it Bedtime.
3. Set the trigger for this event to conditions, the following conditions would be added.

*IF Switch A1 Has been ON for 5 minutes*

4. Set the Actions for this event to turn off all the lights downstairs lights.

This will allow you to leave the room (with the lights on) and have the lights turn OFF 5 minutes later. A modification to this could turn on the bedroom lights when the button was pressed, then turn the downstairs lights off 5 minutes later. Modify the conditions on the bedtime event to trigger by X10 command and trigger on the command A1 ON . Set the actions for this event to turn on the bedroom lights. Create a second event and set its trigger to conditions and set the condition to what we set above: (IF Switch A1 Has been ON for 5 minutes). Then set its actions to turn off all the downstairs lights.

Conditions can also contain an X10 trigger. That is, the condition is dependent on the reception of a particular X10 command.

### **Using conditions to control devices using timed offsets**

Conditions can be used to delay the action of a trigger. Suppose you would like a light to come on at 10:00 PM, and then have a second light go on 30 minutes later. Create an event that triggers at 10:00 PM and turns ON light #1. Create a second event and set its trigger to the following condition:

*If (light #1) (HAS BEEN ON FOR) (:30)*

Set the action for the event to turn ON light #2. You can add as many events as you like as offsets from the light #1 event.

### **About Time After/Time Before Conditions**

Note that using the "before" and "after" time conditions are based on a 24 hour day. For example, if you would like the condition to trigger "after 9:00 pm", the valid range is "9:01 pm to 11:59 pm". This makes sense since 12:00 am starts a new day and is treated as "time 0:00". The condition "before 8:00 AM" is valid between "12:00 AM and 7:59 AM". The "day time" and "night time" conditions make dealing with this issue a little easier.

See the *timers* section to see how timers can be used with devices.

### **Or conditions**

The condition triggers if all the conditions in the condition list are true. It may be desirable to trigger the event if *any* of the conditions are true, rather than all of them. You can create *OR* groups, where all conditions in a particular group must be true, but the event will trigger if any group is true. This allows you to group your *AND* conditions together and *OR* them with other groups. Click the *Next OR Group* button to create a new group.

### 3.15 Trigger Event by EMAIL reception

HomeSeer can check your email mailbox at any interval you would like. It will download your mail and check the recipient address to see if it matches this trigger.

- In the trigger dialog, under type, select *EMAIL reception*. Enter the address of the person you wish to check for. This would be the return address. For example, if you would like a light to go on whenever your wife sends you mail from work and her email address at work is "karen@homeseer.com", enter "karen@homeseer.com". Only email from her will trigger the event. If you want the event to trigger whenever you get ANY email, check the box labeled *all addresses*.

If you set up HomeSeer to check your mail using a POP email account, note the following:

- Check your current email application and see if it is **DELETING** your email from the server after it downloads it. Most readers are set up this way so that your ISP does not complain about you using server disk space. However, if you leave your email reader running at all times, along with HomeSeer, HomeSeer may not see all of your email since your reader is deleting it off the server before HomeSeer can check it. You can do one of the following:
- Don't run your email application unless you are reading your mail
- Change the settings of your email application to not check for mail unless you instruct it to. (Use the send and receive button in your email client to check your mail.)
- Change the settings of your email application to download your mail, but don't delete the mail off the server. You may be able to instruct it to delete the mail off the server when you delete it from your mailbox. (Outlook Express and Outlook 98 will allow this)

**Note that you will not be able to read your email from the web interface if you setup your email to use POP. The ability to read your email from the web interface requires you to use MAPI as your email transport. See the web section for more information on accessing your email via the web page.**

---

# Chapter 4

---

## 4 Event Actions

Every event must have some type of action associated with it. There are two types of actions: *X10 device specific actions* and *global actions*. Global actions are system specific and do not require an X10 device. Actions are defined from the *Event Properties* dialog. Select the tab under the type of action you would like to perform.

1. Select the *Event* view from the View pane.
2. Now right click on an event and select *Event Properties*.

### 4.1 Event Action Order

When an event is triggered, the actions are executed in the following order:

1. Dial an internet connection, or disconnect from the internet
2. Global X10 actions (like All Units Off, etc.)
3. Individual X10 device actions
4. Playing a Sound
5. Launching an Application
6. Send infrared commands
7. Security panel actions
8. Triggering other Events
9. Handle phone actions (if HomeSeer Phone is installed)
10. Speak a phrase
11. Sending email
12. Run a script

#### See Also

Event Actions

### 4.2 Device Specific Actions

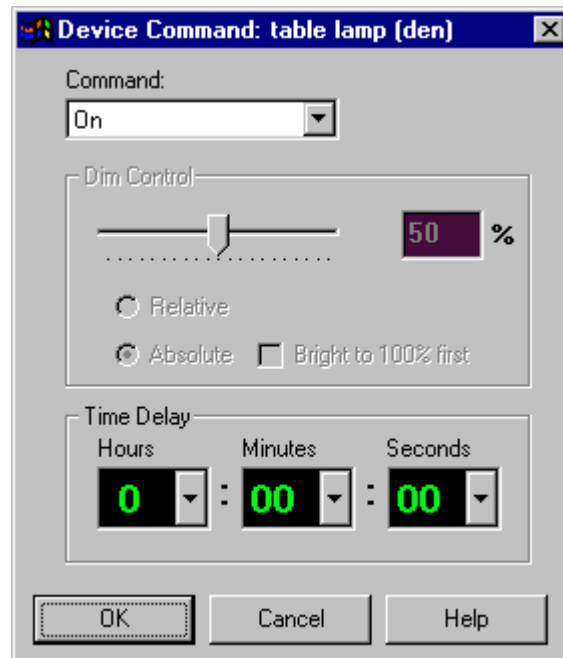
Device actions are set for individual devices. With a single event, you can control many devices and each device can have a different action. Here is how you add X10 devices you wish to control to your event:

1. From the *Events View* select an event.
2. The bottom pane will display any devices already associated with this event.
3. If you want to add a new device, right click in the devices pane (the bottom pane), then select *Add Device...* from the pop-up menu.
4. A dialog will be displayed with a list of all your devices in the upper pane, and the list of currently controlled devices in the bottom pane.
5. To add a device, simply select a device from the upper pane and drag it to the lower one. The device will be added to the list in the lower pane.

- To set or modify the action for the device, double click (or right click) the device in the lower pane. A dialog will be displayed giving you X10 control options.
- Set the X10 command for the device. If you need a delay before the command is executed, enter a delay time from 1 second to 24 hours.

**If the device you wish to control has already been added to the event, do this:**

- Right click on the device, and then select Device Action... from the pop-up menu.
- A dialog appears that allows you to set the action for the device.



**Figure 14 Device Action Dialog**

The following actions are available for the X0 device. Note that different combinations of actions will be available depending on the type of device. For example, devices that do not support dimming will not display the dim control selections.

#### **Absolute Dim**

This will either brighten or dim the light to set it to the desired dim level. It uses the current dim level of the device. Note that the dim level HomeSeer *thinks* the device is at, and the actual dim level, may be different. If the lamp was dimmed locally, HomeSeer would not know the new dim level. For some devices, it may be more desirable to use the *Absolute Dim Bright First* selection.

#### **Absolute Dim Bright First**

This will first brighten the light to 100%, then dim it to the proper dim level. Use this selection when you are not sure what the current dim level of the device will be, or HomeSeer is unable to track the dim level.

#### **Relative**

This simply dims or brightens the light by the dim percent given.

### SmartON

If the SmartOn checkbox is checked, HomeSeer does some extra checks when it comes time to control the device. If the command is an ON, DIM, or BRIGHT command, and the device to be controlled is already ON, then no command is sent. This is useful if you program your living room lights to come on at 60% at dusk, but some days you manually turn the lights on to 100% earlier. Without the SmartOn feature, your lights would dim down 40% after you just turned them on to 100%. The SmartOn feature avoids this nuisance.

### SmartGroup

SmartGroup is used when the event triggers and the X10 commands are sent to your devices. If this option is enabled, HomeSeer will group together all devices with similar actions and located on the same housecode. The end result is that all these devices will react at the same time. For example, if you have devices A1 and A2 both set to bright to 75% at 9:00 PM, and SmartGroup is enabled, both lights will brighten at the same time. If SmartGroup is off, the lights will brighten in sequence, which is not as pleasing. Note that it takes more time to control devices when SmartGroup is on. This is because HomeSeer must send X10 address commands to address each device before it can send the actual bright command.

### Controlling thermostats

If the device you are controlling is a thermostat, you will have different choices available. From the action dialog you can set the setpoint, mode of operation, or control the fan.

Set the action to none, if you do not want any action performed for the fan or the mode. If you are setting the setpoint, enter a new setpoint temperature and check the *Set Temp* checkbox. Leave this checkbox unchecked if you do not wish to set the setpoint.

If you want the action delayed for a specified amount of time. Select a delay time.

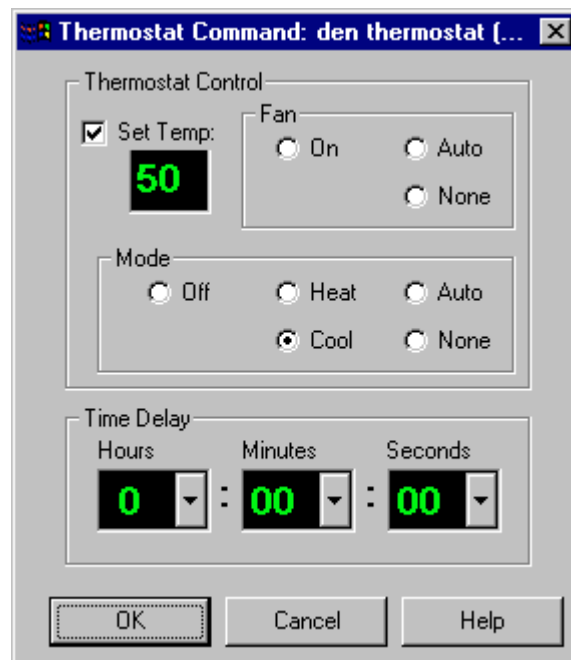


Figure 15 Thermostat Control

### Controlling non-X10 devices

You can also control non-X10 devices using this dialog. If you have some virtual devices set up, you can set their status here. You can also set variables and flags in some home controllers such as the Ocelot controller. If the device you are controlling is a controller variable, then the variable inside the controller will be set when the action is executed. If the device is a variable, the dialog will allow you to enter a value for the variable.

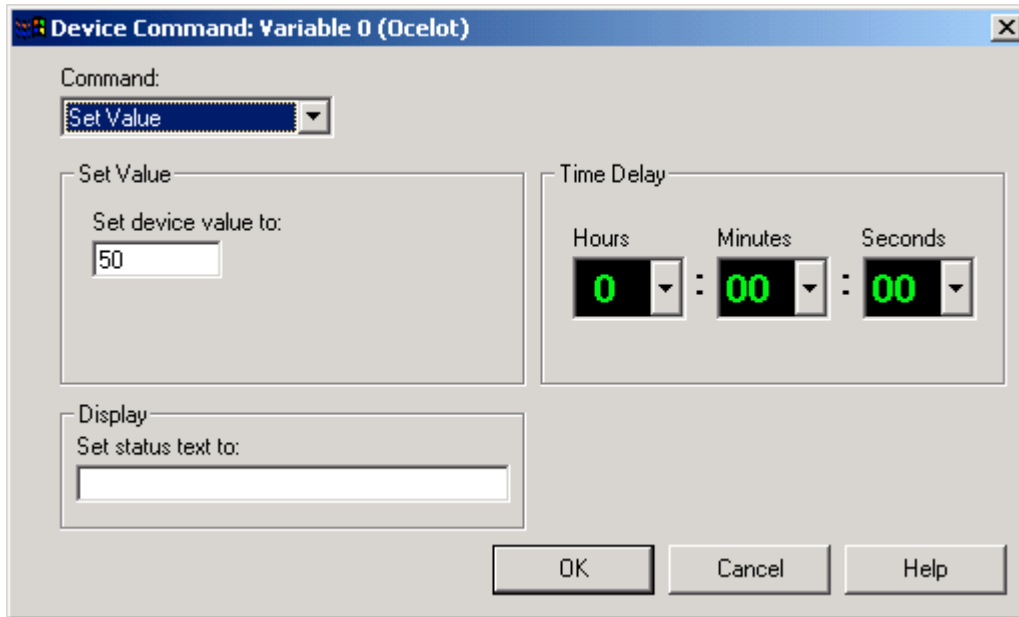


Figure 16 Device Command Dialog

## 4.3 Global Actions

The following global actions are available. These actions are not specific to devices.

## 4.4 Sending housecode wide X10 commands

You can send house code wide X10 commands as an action to an event.

Select the *X10 Device Actions* tab. The lower section of this tab allows you select *global* X10 actions. You may turn ON or OFF all the lights on a particular house code. Select the desired X10 global command from the drop down list box. Not many devices support the *ALL Lights OFF* command. You may use the *ALL Units OFF* command instead. Set the desired *House code* from the drop down list box.. This will turn off all your appliances also. Group all your appliances on a separate house code to avoid this.

If you need to send multiple ALL .. commands, add a device that resides in the housecode you wish to control to the action list. Right click the device to select its action, then select the ALL ... command you wish to send. You can add as many devices as you like.



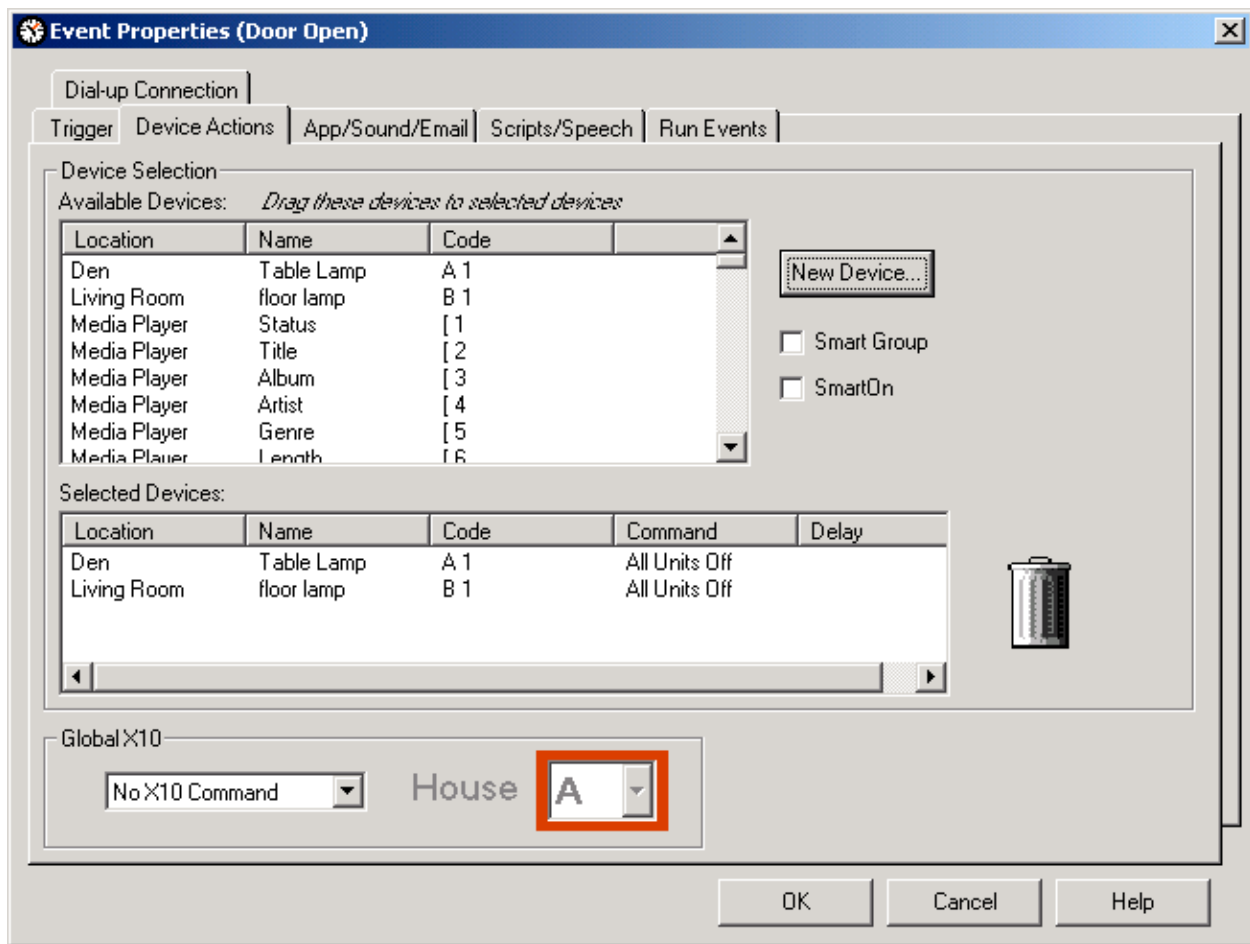


Figure 17 Sending multiple Housecode wide commands

## 4.5 Launching an Application

You can launch an application as an action to an event.

Select the *App/Sound/email tab* from the *event properties* dialog. Select the application you wish to run by clicking the *Browse...* button. A dialog will be displayed that will allow you find the application you wish to launch. If the application requires any parameters, you can enter them in the parameters textbox.

If you have an application that will dial the telephone, you can launch this application to send a page.

## 4.6 Playing a Sound

You can play a sound as an action to an event.

Select the *App/Sound/email tab* from the *event properties* dialog. Click the *Browse...* button to display a dialog that will let you find a WAV file to play when the event is triggered. The "media" directory in your system folder (c:\windows\media) is a good source of WAV files. You can test the sound by clicking the play button (*the small triangle next to the browse button*).

## 4.7 Sending email

You can have an email sent out as an action to an event.

Select the *App/Sound/email* tab from the *event properties* dialog. If you have an Internet connection, you can have HomeSeer send email when an event is triggered. This may be useful if you are away from home and HomeSeer is monitoring your security system. It can send you email if the alarm went off, your motion detectors sensed motion, or whatever you like. Enter the subject text you would like in the message and the text you would like in the body of the message. HomeSeer will always include the reason the EMAIL was sent in the body of the message. You can set a default subject under the menu *View->Options->EMAIL*. See *Setting up email* for more information on setting up email. You can also send attachments with the email. Add the path to the file to be attached. Attachments can be useful if you use a video camera to snap pictures. You can then have HomeSeer email you the picture at particular intervals. If you add \$ip to the message body of the notification, HomeSeer will insert the IP addresses of all your network adapters on your system. This can be handy if you use a dial-up connection and receive a different IP address every time you connect. You can have HomeSeer email you its IP address when it dials into the Internet.

To send email to multiple addresses, separate each address with a comma.

## 4.8 Dialing an Internet Connection

An event can be set to dial the phone and make an Internet connection. To create an event that dials the Internet, do this:

1. Create a new event can call it something like Dial Internet
2. Set the trigger to the time you want your computer to dial the internet. Recurring might be a useful trigger, as you may want the computer to dial up on the hour so you can check your home from a remote location.
3. On the *Misc Actions* tab in the event, click the drop down list labeled *Select Connection*, and select the Internet connection you want to dial. Tip: Make sure this connection has a username and password saved with it, otherwise the computer will prompt for this, which is not good if the computer is unattended.
4. Select the number of times the computer should retry the connection if it fails, and the interval for retrying.
5. If you would like to disconnect from the Internet after a number of minutes, enter the number of minutes in the *Disconnect after # minutes* box.

## 4.9 Running a script or speaking a phrase

You can have HomeSeer run a special script or speak a phrase as an action for an event.

### Scripting

Select the *Scripting/Speech* tab from the *event properties* dialog. Here you may select the script or scripts that you wish to run. The list box will display all available scripts. Select the script you wish to run, then click the *Add Selection* button. The script is added to the text box. You may enter as many scripts as you like. A comma must separate each script in the list. The scripts will be executed in the order that they appear on the line. You can also edit scripts from here. Select the script you wish to edit from the list box, then click the *Edit* button. Whichever editor is configured to edit .txt files will be loaded and the script displayed. You may test the script by clicking the *Test* button after you have added a script to the run box. If multiple scripts are in the run box, only the first script will be run when the *Test* button is pressed.

If you want to create a new script, click the *New* button. You will be prompted for the script type. Enter the script name and the appropriate extension. The following script types are supported:

- .txt files are VBScript scripts
- .js files are JavaScript scripts
- .pl files are PerlScript scripts

**Don't forget to save your script before you test it! Any errors that were found by the scripting engine will be logged to the event log.**

### Script Parameters

Scripts can also be called with parameters. The format is:

```
Filename(function,parameter)
```

Consider the following script named `dim_bright.txt`. The script will dim any light to 0% or brighten to 100% without adding the actual device code to the script. The device code can be passed.

```
` the main function is not used

Sub main()

End sub

Sub dimlight(device)

    hs.ExecX10 device,"dim",100,0

End sub

Sub brightlight(device)

    hs.ExecX10 device,"dim",100,0

End sub
```

To call this script from the event properties, you add the function and parameter to the script filename. To dim the light "A1", the format would be:

```
dim_bright("dimlight","A1")
```

To brighten the light B2, the format would be:

```
dim_bright("brightlight","B2")
```

### Script Statements

You can also run script statements. If all you need to do is execute a single statement or IF-THEN-ELSE logic, then this method avoids the overhead of creating a text file for the script. Script statements must be preceded with a "&" character to tell HomeSeer to just execute the given statement.

For example, suppose you just wanted to change the displayed status for your door sensor so that it displayed the text *Door Open*, as opposed to *ON*. You can set the display text for a device with the script statement `hs.SetDeviceString()`. Just enter the following statement in the script run box:

```
&hs.SetDeviceString "A1","Door Open"
```

Another example:

Suppose you wanted to turn the light A1 off if device B1 was on, else speak. You could do this with a single IF statement like:

```
&if hs.IsOn("B1") then hs.ExecX10 "A1","off",0,0 else hs.Speak "B1 is not on"
```

Note that you cannot add multiple line scripts in the script run box. You will have to create a script file for this. You can however add multiple script commands. Separate each command with a colon. For example:

```
&hs.speak "the door is open":hs.SetDeviceString "A1","Door Open"
```

### Avoiding running the same script multiple times

If an event is triggered repeatedly, the same script may be running multiple times. This is normally not desired, depending on what the script is doing. If the script is accessing the web to download some information, you only need one copy of it to be running. On the scripting tab, there is a checkbox:

```
Do not allow multiple copies of the script(s) to run
```

This is checked by default. If checked, only one instance of the script can run at a time. If you need the script to be executed every time the event is triggered, then uncheck this box.

### Speaking

From the Scripting/Speech tab you can add some text that will be spoken when the event is triggered. Enter the string you wish HomeSeer to speak. You can test the string by clicking the *Test* button. There are some special strings you can insert into the phrase that allows HomeSeer to speak things like the date and time. They are:

- \$time (says the current time)
- \$date (says the current date)
- \$from (speaks the address of the last email received. If the trigger was caused by an email received, this speaks the address the email was from)

For example, the following phrase would tell you the time and date:

```
The time is $time and the date is $date
```

If *Speak in background* is not checked, speaking will hold up other actions in the event until the speaking has stopped. Normally, you should check this box so the speaking is done in the background and other events are not held up.

### Controlling hardware right before and right after speaking

It may be desirable to control some type of hardware just before the computer is ready to speak. For example, maybe you need to route the audio to a specific set of speakers. You can run a script a command or run an entire script just before the computer speaks, and right after speaking is finished. Enter the script commands or script filename in the proper text boxes on the Speak tab.

**Figure 18 Running scripts before/after speaking**

## 4.10 Triggering other Events

You can run other events as an action to an event.

Select *Events* tab from the *event properties* dialog. From this dialog you can force the execution of a different event. This is useful if you create an event that controls many devices, such as setting the lighting for a movie. You may want to trigger this event in many ways. Simply add the event name to the *Run Order* list. The list box on the left shows all your events.

1. Select the event you want to run, and then click *Add*.
2. The event is now displayed in the right list box. The right list box lists all the events that will run. You can enter as many as you like.
3. The events will be run in the order they are listed.
4. You can change the order the events are run in by clicking the *Up* and *Down* buttons.
5. Click the *Delete* button to remove events from the list.
6. Check *All conditions applied to the event must be true* if you only want the events to be run if their conditions are true. If this box is unchecked, the actions to the events will execute regardless of their trigger type. This is useful if the event to be executed is to be treated as a macro, and its trigger is not relevant.

**Be careful that you do not select the current event without checking its conditions. This would cause a loop where HomeSeer would be continuously triggering the same event.**

## 4.11 Sending an infrared command

You can cause infrared commands to be sent to your audio/visual equipment as an action to an event.

Select *I/R* tab from the *event properties* dialog. Here you can send an infrared command to your audiovisual devices. See chapter 6 for more information on setting up your I/R devices. Setup an I/R command sequence as follows:

1. Select the device/command you wish to control (TV ,on for example), then click the *Add Device* button
2. The text *tv,on* will be added to the command text box.

3. The *command* text box will display all your commands. Each command will be separated with a comma.

You can mix devices in the command text box.

You can test your custom command string by clicking the *Test* button. This will send the command to the infrared device.

#### **Delay Sending a command**

If the infrared command is being sent to quickly to your equipment, delays may be inserted into the command string. To insert a delay, add an empty ", ". Each delay is 1/4 of a second. For example, to change the channel of the television, but insert a 1/2 second delay between the channel digits, use the following command:

```
tv,1,,,2
```

#### **Voice commands and infrared**

Voice command responses can be used as an input to IR commands. The variables \$# can be used to substitute words from a voice command into the send I/R line. See the infrared control section for more information.

### **4.12 Infrared Match Trigger**

Some infrared devices, like the CPU-XA/Ocelot and JDS IRXpander, support infrared matching. The device will notify HomeSeer if it sees an infrared command that has been previously learned. You can set up HomeSeer to trigger on these matches. If the selected infrared device supports this feature, then the infrared match selection will be available.

This feature is useful if you would like to know when you power on your stereo. You could then change the lighting or perform some other action.

Note that the Applied Digital Ocelot can only match on the first 80 Infrared commands.

### **4.13 Security Panel Trigger**

HomeSeer includes support for the Napco Gemini security panel using the HAWizard control or the HAI Omni using the HAI plug-in. To configure a panel trigger, select the proper panel trigger from the trigger list. See the HAI documentation for details on setting up triggers.

### **4.14 Security Panel Actions**

An event can perform some actions to your security panel. Different panels will have different options available. To set the actions, click the proper action tab in the event properties.

Refer to your Napco Gemini or HAI documentation for more information about the actions that the panel can perform.

### **4.15 Timers**

HomeSeer has built-in timers for ALL devices. The time since the device last changed state is kept up to date. For example, when HomeSeer starts, the timers for all devices are reset to 0. When the state of a device changes, like from On->Off, the timer is reset to 0. You can use these timers in the conditions to see how long a device has been ON or OFF. This is useful for turning off lights that have been turned on via a motion sensor. You can use the condition:

*IF light has been on for at least 10 minutes then (turn it off)*

As motion is detected, the status of the light will be reset to ON, and the timer will be reset to 0. So the light will stay on as long as there is motion, and will turn off after motion as stopped for 10 minutes.

Script commands can be used to query any timer. The following command will retrieve the time since last changed for the device A1:

```
mytime = hs.DeviceTime("A1")
```

HomeSeer also saves the absolute time of the last status change. This time includes the date and time of the change. You can see the absolute time in the device list. To access this, you can use a script command. The command is:

```
Last_change = DeviceLastChange("A1")
```

See the scripting section for more information about HomeSeer scripts.

## 4.16 Groups

Each event may be assigned to a specific group. The group is simply a name and has no other significance. This allows you to put all related events into the same group. For example, you may have a bunch of events that control your motion sensors. You could put them all in your *motion sensor* group. Putting all your voice commands into their own group is also handy.

There is no limit to the amount of groups you can create.

To move a collection of events to a new group, hold down the *control* key while clicking each event you want to move. This will highlight each event as it is selected. After selecting all the desired events, *right click* one of the them and select *Move to Group*, then *New Group* from the pop up menus. A dialog box will appear asking for the name of the new group. Enter a name for the group, then click *ok*. All the events will be assigned to this new group.

You can view a selected group by clicking the group name from the drop down list at the top of the event list.

Selecting *All Groups* from the drop down list will display all of your events.

The group's list is generated by the names given to groups. There is no need to manually create or delete groups as this is done automatically.

## 4.17 Security

Checking Security Mode from the Tools menu may enable security mode. You can mark events as participating in security mode by checking the checkbox *Security* in the event properties. When security mode is enabled, the event trigger time is randomly set to plus or minus 30 minutes (settable) from the actual set time. This can give your home a lived in look.

Note that only the following trigger types support the security mode:

- Absolute time trigger
- Sunrise/Sunset with offset trigger
- Absolute time using the conditions
- Conditions based on time

Note that the current time offset for the event is displayed next to the event trigger in your event list. If the offset was randomly set to +20 minutes, the event trigger will be followed by:

(Sec: 20)

The offset time can be set from the *General* tab in the *Options*.



# Chapter 5

## 5 Text to Speech Support and Voice Recognition

HomeSeer includes Text-to-Speech with or without Microsoft Agent.. When you Install HomeSeer, all the proper files are installed for MSAgent text-to-speech, and voice recognition. If you do not wish to use either one of these features, you can disable them in the HomeSeer options.

HomeSeer supports two different text-to-speech interfaces; SAPI4 and SAPI5. Microsoft Agent uses the older SAPI4 text-to-speech. Microsoft has released a newer SAPI5 text-to-speech interface. SAPI is simply a programming interface. Text-to-speech engines (voices) normally support one (or both) interfaces. There are many voices available for SAPI4, so HomeSeer continues to support this interface. Any SAPI4 compatible voice can be used with Microsoft Agent. The agent is not compatible with SAPI5, so even though HomeSeer supports SAPI5, the voices will not be spoken through the agent character.

HomeSeer Voices, which are provided by AT&T™, are now available These high quality voices are among the best available and can be purchased as an extra cost add-on for HomeSeer. Visit [www.homeseer.com](http://www.homeseer.com) for voice samples and more information. The HomeSeer Voices support both SAPI4 and SAPI5 text-to-speech interfaces so they may be used with and without Microsoft Agent.

If you are not sure what MSAgent is, you might want to take a quick tour at:

<http://www.msagentring.org/>

MSAgent is a technology that uses interactive characters that sit on your desktop. You can use them to speak and to listen. Microsoft is making MSAgent part of its operating system. It is included with Windows 2000.

When you install HomeSeer, MSAgent is installed automatically. You do not have to use the agent characters if you don't want to. To disable it, use the text-to-speech tab in the options. From this same tab, you can select the character you want to use. HomeSeer includes one character, the genie. You can get many more characters from the web, just install them and return to this dialog to select them. A good source for agent characters is at:

<http://www.msagentring.org/>

### 5.1 Voice Recognition

HomeSeer listens for an attention phrase, then acknowledges the attention phrase, and then listens for commands. You must speak the attention phrase first. If you do not, commands will be ignored. Go to the VR options tab from the menu *View->Options->Voice Recognition*.

1. Check the checkbox *Enable*.
2. Enter an attention phrase, such as *computer*. You will need to say this phrase whenever you want the computer to listen for your voice commands. The computer will respond with an acknowledgement and will start listening for commands. Be careful what you enter for the attention phrase, as the computer will always be listening for this and may hear this phrase in background noise. If you set an ignore phrase, the computer will continue to listen for commands until it hears the ignore phrase. This allows the following conversation:

[you] computer

[computer] yes sire

*The computer is now listening for all your voice commands*

[you]turn on the table lamp

[computer]turn on the table lamp

[you] turn on the TV

[computer] turn on the TV

[you (speak the ignore phrase)] goodbye

[computer (speaks the ignore acknowledge phrase)] goodbye sir

*The computer now goes back to listening only for the attention phrase*

3. You can enable another mode where the computer will ask for confirmation for each voice command. Check the box *commands require confirmation* and the computer will ask you if it should execute the voice command. Confirmation can also be enabled or disabled for each voice command. To enable confirmation on a particular voice command, check the box *Confirm* in the specific event's properties, trigger tab.

Using scripts, you can interact with your computer. Through the scripting interface you can add your own voice commands that are active for as long as the script is running. See the *messages.txt* script and the *stock\_quotes.txt* script for an example. The messages script will tell you how many email messages you have in your inbox and ask you if you would like them read to you. The stock\_quotes script will retrieve stock quotes from the Internet. See the scripting section for more information on the scripting interface.

Acknowledge phrases may text or a WAV file. If you would like your acknowledge phrase to be a WAV file, enter the complete path to the file in the *Acknowledge Phrase* text box.

## 5.2 Attention Phrases

Attention phrases may be a single word, or a group of words. Try different words to find the one that works best for you. Simple words are not recommended as they can cause false recognitions. You can allow optional words by using "[" around the words. For example, suppose you wanted the computer to respond to "computer" OR "hey you". The attention phrase would be:

[computer] [hey you]

Note that HomeSeer automatically raises the recognition confidence level of the attention phrase. This helps to keep the computer from thinking it heard the attention phrase when background noise is present. The confidence level is raised by prepending a '+' character to the beginning of the attention word.

### Controlling when HomeSeer is listening for attention or commands

After you give the attention phrase, HomeSeer is then awaiting a voice command. After you give a voice command, HomeSeer then goes back to listening for the attention phrase. If you want to continue to give voice commands, set an ignore phrase in the options. HomeSeer will then continue listening for commands until the ignore phrase is heard. When it hears the ignore phrase, it will return to listening for the attention phrase only. There is also a timeout associated with the ignore phrase. The timeout default is 30 seconds. If no command is given after 30 seconds, it then automatically returns to listening for the

attention phrase. The timeout is set in the options and must be in the range 1 to 60 seconds. This timeout can also be disabled. This timeout is useful for the case where an attention phrase was accidentally detected and no one was around to cancel it. False triggers are thus avoided.

### 5.3 Intercepting the Attention Phrase

It may be desirable to intercept a voice command before it can be processed. For example, you may have a voice command that includes all members of your family. You can intercept the attention phrase and set the voice recognition speaker for the desired individual. This is accomplished through the use of a special script. If you have a script with the filename:

```
check_vcmod.txt
```

This script will be run before the voice command is processed. The recognized voice command can be accessed by calling:

```
hs.LastVoiceCommand
```

If the computer did not recognize the spoken phrase, the string returned by the `hs.LastVoiceCommand` call would return a "\*", which indicates an unrecognized phrase. You can use this to speak "I don't understand". The example script below demonstrates this.

If you would like to have the computer ignore the recognized phrase, set `hs.LastVoiceCommand` to "\*". This will tell the system to ignore the command and not act on it.

For example. Suppose you had an attention phrase that allowed you to access the computer with a different name. Two users called the computer by name, one uses "jack", the other uses "roxy". The attention phrase would be:

```
[jack] [roxy] or (jack|roxy)
```

The `check_vcmod.txt` script would look like:

```
sub main()

  dim s

  s=hs.LastVoiceCommand

  if s="jack" then
    hs.setspeaker "dad"      ' dad is talking
    hs.speak "hello dad"
  end if

  if s="roxy" then
    hs.setspeaker "mom"      ' mom is talking
    hs.speak "hello mom"
  end if

  if s="*" then
    hs.speak "I don't understand"
  end if
end sub
```

## 5.4 Formatting Voice Commands

There are some tricks you can use to format voice commands. Consider the following prople:

You would like to give a command that changes your television to a specific channel.

You could do this by creating a voice command for every possible channel, but that's impractical. Instead, your voice command can contain ranges of words. The recognition engine will accept any one of the words. To create a command that changes your television to any channel would look like this:

```
TV channel (0|1|2|3|4|5|6|7|8|9)+
```

This will accept any command like "tv channel 0 1", or "tv channel 1 3 6". You can substitute the actual word for the number in the command like:

```
tv channel (zero|one|two) etc.
```

It works the same either way.

**Note that nested “()” and “[ ]” are not allowed. The following will not work: (hello|(bill|sue))**

See the voice command section in chapter 6 for information on how to control IR devices with voice commands.

The string expression you supply can include square bracket characters ([ ]) to indicate optional words and vertical bar characters, (|) to indicate alternative strings. Alternates must be enclosed in parentheses. For example, "(hello [there] | hi)" tells the speech engine to accept "hello," "hello there," or "hi" for the command. Remember to include appropriate spaces between the text that is in brackets or parentheses and the text that's not in brackets or parentheses. You can use the star (\*) operator to specify zero or more instances of the words included in the group or the plus (+) operator to specify one or more instances. For example, the following results in a grammar that supports "try this", "please try this", "please please try this", with unlimited iterations of "please":

```
"please* try this"
```

The following grammar format excludes "try this" because the + operator defines at least one instance of "please":

```
"please+ try this"
```

The repetition operators follow normal rules of precedence and apply to the immediately preceding text item. For example, the following grammar results in "New York" and "New York York", but not "New York New York":

```
"New York+"
```

Therefore, you typically want to use these operators with the grouping characters. For example, the following grammar includes both "New York" and "New York New York":

```
"(New York)+"
```

Repetition operators are useful when you want to compose a grammar that includes a repeated sequence such as a phone number or specification of a list of items:

```
"call (one|two|three|four|five|six|seven|eight|nine|zero|oh)*"
```

```
"Id like (cheese|pepperoni|pineapple|canadian bacon|mushrooms|and)+"
```

Although the operators can also be used with the optional square brackets grouping character, doing so may reduce the efficiency of Agents processing of the grammar.

You can also use an ellipsis (...) to support *word spotting*, that is, telling the speech recognition engine to ignore words spoken in this position in the phrase (sometimes called *garbage* words). When you use ellipses, the speech engine recognizes only specific words in the string regardless of when spoken with adjacent words or phrases. For example, if you set this property to "[...] check mail [...]", the speech recognition engine will match phrases like "please check mail" or "check mail please" to this command. Ellipses can be used anywhere within a string. However, be careful using this technique as voice settings with ellipses may increase the potential of unwanted matches.

When defining the word grammar for your command, include at least one word that is required; that is, avoid supplying only optional words. In addition, make sure that the word includes only pronounceable words and letters. For numbers, it is better to spell out the word than use an ambiguous representation. For example, "345" is not a good grammar form. Similarly, instead of "IEEE", use "I triple E". Also, omit any punctuation or symbols. For example, instead of "the #1 \$10 pizza!" use "the number one ten dollar pizza". Including non-pronounceable characters or symbols for one command may cause the speech engine to fail to compile the grammar for all your commands. Finally, make your voice parameter as distinct as reasonably possible from other voice commands you define. The greater the similarity between the voice grammars for commands, the more likely the speech engine will make a recognition error. You can also use the confidence scores to better distinguish between two commands that may have similar or similar-sounding voice grammar.

## 5.5 New SAPI5 changes to voice command formats

This version of HomeSeer now supports the SAPI5 programming interface for voice recognition. This required that the SAPI4 voice command format to be translated to XML, as SAPI5 uses a new XML format for grammar. Every effort has been made to insure that voice commands created with previous versions of HomeSeer will work with the new version. There are some exceptions.

HomeSeer converts all the voice commands and attention phrases to XML and creates two XML files:

1. grm\_attention.xml = XML file holding attention phrase.
2. grm\_command.xml = XML file holding all active voice commands

These XML files are loaded by the recognition engine when listening for the attention phrase or for commands. If your voice recognition is not working, check the event log for any error messages, as there could be an error in one of the XML files. A dialog will also display when an error is detected.

The following exceptions should be noted when creating voice commands

- The '+' character is used to denote a phrase that can be repeated on or more times. The character must be present after the word. If it's present before the word, like "turn +off the lights", then it indicates that the recognition confidence of the word should be raised.

## 5.6 Using Dictation with Voice Recognition

There are times when it might be nice to allow any word to be recognized in a phrase. Consider the task of creating a recognition phrase that allows you to add items to a grocery list. Without dictation, you would need to create a phrase for every item you might want to add to the list. Using dictation, you can speak any word. To mark a word as a dictation word, use the tag: <DICTATION/> Here is a phrase that will add any item to a grocery list:

Add <DICTATION/> to the grocery list

Note that the above command will only allow a single word to be added to the list. To add multiple words, use:

Add <DICTATION/>+ to the grocery list

See the *shopping\_list.txt* script for an example. This script can optionally use dictation.

Note that you will probably need a 800mhz PC or faster to use dictation as it requires a fast CPU.

## 5.7 Creating Voice Commands to control devices

See the section *Creating/Editing devices* for instructions on how to create voice commands that will control devices directly.

## 5.8 Troubleshooting Speech Recognition

You want to make sure that you have a good quality microphone. Also, test your sound card by recording your voice with the sound recorder that is available on your Accessories start menu. If you can record your voice, speech recognition should work.

If the computer refuses to respond to your voice commands, check the following:

1. Re-run the *microphone setup wizard*. To do this, bring up the speech control panel from your control panels, then select *Configure Microphone* button.
2. Make sure the voice recognition is enabled by checking the *Enable* checkbox on the *voice recognition* tab in the options. Restart HomeSeer after enabling this checkbox.
3. Check the status bar at the bottom of the main window. It will tell you if HomeSeer is listening for the attention phrase or listening for commands. You can always say the attention phrase even if HomeSeer is listening for commands. Press Control-L to start HomeSeer listening. By default, the attention phrase is *Computer*.
4. Check your event log for startup error messages. An error message will also be logged for events that have improperly formed voice commands.
5. Not all sound systems work properly with voice recognition. Some older 8 bit sound cards will not work at all. You need a good quality 16-bit sound card. If the recognition is not working at all, or is very unreliable, try installing a newer sound card. Sound Blaster cards work fine. If you cannot record your voice clearly with your sound recorder, then recognition probably will not work.
6. Most laptop computers will not support voice recognition, especially older ones. Newer ones may work if you use an external microphone.
7. Microphones built into the monitor or laptop usually do not work.

## 5.9 Using MSAgent

Here are some tips for using MSAgent:

- You can configure MSAgent by opening the properties dialog from the tools menu: *Tools->MSAgent-Properties*  
From here you can adjust how fast the agent speaks, etc.
- You can have the agent hide after he speaks by enabling *View->Options->Text-to-Speech->Hide agent after speaking*. Note that to give a voice command while the character is hidden,

you must press the voice command key (scroll lock by default), and then say the character name. You will have to press the voice command key a second time to have it start listening.

- If the agent is making noises such as snoring while its idle, disable the agent sound affects from the MSAgent properties from the tools menu.

---

# Chapter 6

---

## 6 Infrared Control Overview

HomeSeer includes support of the following devices:

- SmartLinc (now owned by SmartHome.com) PCIRLinc model 1195 (serial port device, all OS's supported, single zone, learning)
- SmartHome IRLinc model 1623PC (serial port device, all OS's supported, single zone, learning)
- SmartLinc HouseLinc (serial port device, all OS's supported, multiple zones, non-learning)
- Nirvis Slink-e (serial port device, all OS's supported, multiple zones)
- ConceptUK PCRemote (parallel port device-Win95/98, single zone, learning)
- CPU-XA/Ocelot (serial port device, all OS's supported, single/multiple zones, learning)
- JDS IRXpander (serial port device, all OS's supported, 4 zones, learning)
- PCIR from ziplabel.com (parallel port device-Win95/98, single zone, learning) (this is a "build it yourself" type of device) (Select ConceptUK PCRemote if you have one of these)

All of the above devices are infrared learning devices, with the exception of the HouseLinc, and can learn infrared commands from your existing remotes.

The CPU-XA/Ocelot controller can support multiple zones using the SECU16-IR module. On the IR configuration dialog, you can select the zone that the a particular device is in.

The parallel port devices support up to 32 devices with 36 keys per device. This is a total of 1152 infrared commands.

The CPU-XA/Ocelot supports either 512 or 1024 commands. HomeSeer will detect which unit you have and only display the proper number of devices.



## 6.1 PCIR/Linc

To setup the PC/IR Linc, connect it to an available serial port on your computer. Tell HomeSeer what serial port you are using in the *View->Options->Interfaces* dialog. Use the drop down list labeled *COM Port/LPT Port*, and select the communications port you are using. You will then need to restart HomeSeer. When HomeSeer restarts, check your event log for any error messages. HomeSeer will attempt to communicate with the PCIRLinc when it starts.

Now you must tell the PC/IR linc about what types of audiovisual equipment you are controlling.

Enter the infrared configuration dialog from the menu *Tools->IR Config ...* Click the Options button. A new dialog will display allowing you to enter the device types for your equipment.

Enter the device code in the Code textbox for each device that you are controlling. The device code tells the PC/IR linc what type of infrared commands should be sent for the specific device. This is the same code you entered when you set up the hand held remote for the equipment. The codes for the supported device can be found in the back of the PC/IR linc manual. Assuming your TV was an RCA, you would enter 013 in the code textbox.

After setting the device codes, click the Program device codes button. This programs the information into the PC/IR Linc. Click OK to close this dialog.

You may change the labels for each device, if you desire. If your TV is an RCA, for instance, you may change the name to RCA.

If you are not using a device, leave the name blank. This will keep this device from appearing in other areas of the program.

## 6.2 Applied Digital CPU-XA/Ocelot

This device can be used for X10 and infrared. You may use just the infrared features of this device if you like and use the CM11A for X10. In the options section, Interfaces tab, select the CPU-XA as your infrared device. Also select the serial port number that your CPU-XA/Ocelot is connected to.

There are two models of the CPU-XA/Ocelot. One that supports 512 commands and one that supports 1024. HomeSeer will detect which model you have and enable either 14 or 28 devices depending on the model.

This device does not have any built-in codes for devices, so you must learn all your commands from your existing remotes.

## 6.3 JDS IRXpander

The IRXpander is a serial port device that supports 4 zones of IR and also allows IR to be repeated from its IR input to all IR outputs.

Select the IRXpander from the drop down list of IR devices on the interfaces tab in the HomeSeer Options. You can now assign keys to IRXpander IR locations from the *IR Config* dialog from the *View* menu.

## 6.4 PCRemote/PCIR

To setup these devices, connect it to an available parallel port on your computer. Tell HomeSeer what parallel port you are using in the *View->Options->interfaces* dialog. Use the drop down list labeled *COM Port/LPT Port*, and select the parallel port you are using.

These interfaces do not have any built-in codes for devices, so there are no device codes to give it. Simply create labels for each of your devices, then go to the individual device screens (using the Labels drop down list), and learn the commands from your current Remote.

## 6.5 Slink-e and HouseLinc

Slink-e and HouseLinc support is provided through the use of a Plug-In. Select the appropriate device from the drop down list of IR devices on the Interfaces tab in the options. Select the *Options* button then the *Help* button to get more information on installing and using these devices.

Note that if you are using the Slink-e device, you need to first install the proper software from Nirvis first. This includes the Slink-e server software.

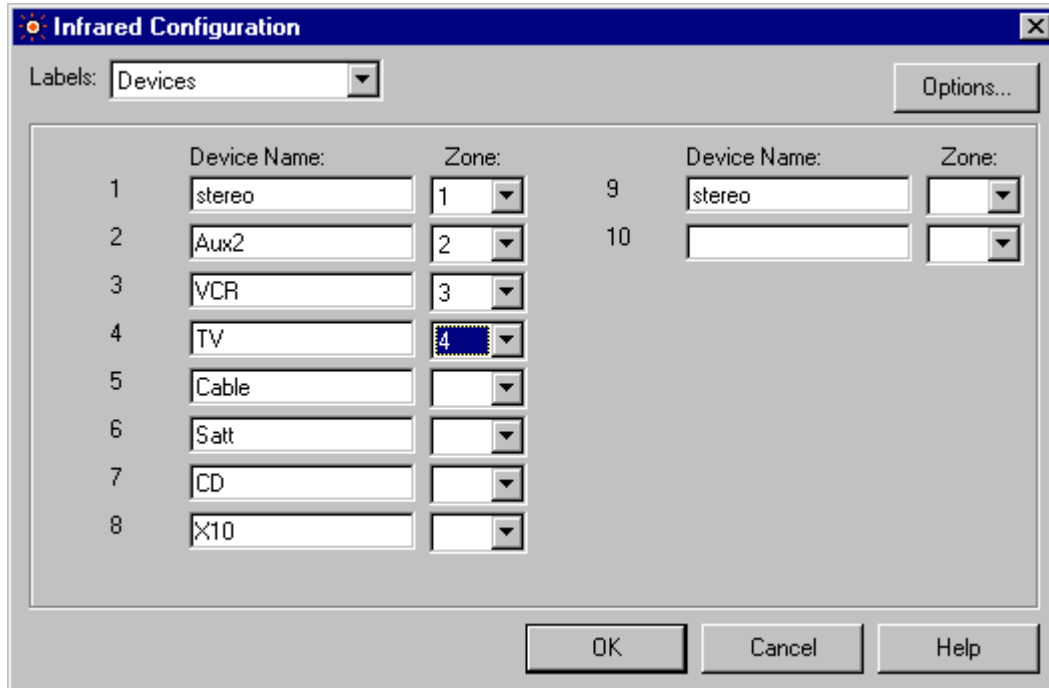


Figure 19 Infrared Configuration Dialog for the JDS IRXpander

## 6.6 Learning I/R Commands from Existing Remotes

If the infrared device you have selected supports infrared learning, the test and learn controls on the bottom of the device labels page will be enabled. If they are disabled, your interface does not support infrared learning.

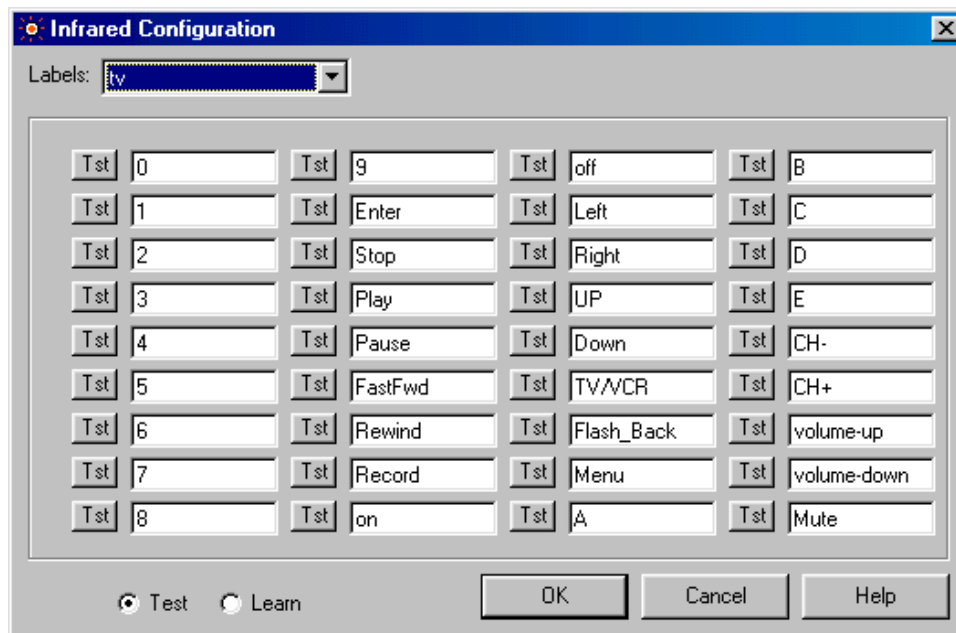
To learn commands from your existing remote:

1. Bring up IR dialog from *View->I/R Config*.

2. Select the device that will receive the new infrared command from the drop down list labeled *Labels*.
3. Click learn at the bottom of the screen. All the keys that are learnable for the device will be enabled and their label will change to *Lrn*.
4. Click the *Lrn* button next to the command you wish to change. Note that not all keys can be programmed. Keys you cannot program are grayed out.
5. In about 5 seconds a dialog will appear. It will ask you to point your existing remote at the learn window on your infrared device. Press the key you wish to learn. If you are learning with the PCIRLinc, hold the key down until the green light on the PC/IR Linc goes out. With other devices, the system will beep after the command is learned. Read the displayed dialog carefully. Some devices require you to click OK then learn, others require you to press the OK button after you learn.
6. Now press *Ok* on the dialog to complete the learning. Checking the Test Mode check box and clicking the Tst button next to the command that was just learned can test the command. *Please refer to your hardware's user's manual for more information on Learning commands.*

## 6.7 Changing I/R Device Key Labels

Default device names and labels are provided for the first 8 devices. If you like, you can enter your own labels. From the View->I/R Config dialog, use the drop down list labeled Labels to select the device you wish to configure.



**Figure 20 I/R Labels Dialog**

This dialog displays all the commands available for the selected device. You can now enter your own labels, if the actual function of the key is different from the default label. If you want to test a key, check the box labeled Test Mode at the bottom of the screen. This changes all the key labels from *Lrn* to *Tst*. When you click a button, it will send the I/R command to your infrared interface hardware. See the following section for more information on learning commands from your existing remotes.

## 6.8 Creating Voice Commands to Control I/R Devices

As mentioned in the voice commands section in chapter 4, you can create voice commands that can control your I/R devices, such as changing the channel on your television. The command would look like this:

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

The "+" above says the preceding string can be repeated one or more times. This means you can say:

```
tv channel 1
tv channel 2 3
tv channel 1 5 7
```

Now we need a way to get the recognized voice command to be sent to our infrared device. The trick is to extract the words from the recognized voice command, match them with our set IR keys, then send them out as IR. There are a couple of ways to accomplish this.

- Make sure the button names for your number keys match what is in the command. In this example, we are using numbers (like 0,1,2 etc.). In the I/R Config dialog, make sure your device number keys are defined with these numbers, by default they are 0, 1, 2 etc. If you are going to send any other type of I/R command, you must use the exact text as described for the key.
- The easiest way to get the voice command to your IR device is to use some variables. The variables \$# can be used to extract the words in a recognized voice command. For example, let's assume the voice command is:

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

We already have an IR device named tv and we have learned IR commands for the device in keys labeled 0 through 9. When you speak the command to change to channel 5, the recognized voice command will be:

```
tv channel 5
```

In the event properties for the event that contains the above voice command, we would set the action of the event to send the following IR command: (The IR tab in the event properties)

```
tv,channel,$3
```

The \$# variables hold the individual words contained in the recognized voice command. In this example, \$1="tv", \$2="channel", and \$3="5". So you can see that by putting \$3 in our IR command, we can send different IR commands based on the spoken phrase.

The above example only handles single digits, but we already know that we can speak multiple digits because we added the "+" after the voice command. This allows us to speak multiple numbers. Subsequent numbers can be represented with \$4, \$5, etc. To handle up to three digits, we can change the IR command to:

```
tv,channel,$3,$4,$5
```

- A second option to handle the conversion of a voice command to IR, is to use a helper script. The script simply retrieves the last voice command and then calls the SendIR method to execute it. See the *Creating Scripts* section in chapter 7 for more information on scripting. The script for

our sample would look like this:

```
sub main()
  dim s

  ' get the last voice command recognized
  s=system.lastvoicecommand

  ' send the command to the PC/IR Linc
  ' note that extra words in the command are ignored
  ' make sure that there are not any words in the voice command
  ' that map to a remote control key
  ' we need to add the device we are going to control here "tv" This
  ' is the same name that was added to the device in the I/R config
dialog
  hs.SendIR "tv,"+s

end sub
```

Note that the voice command will contain other words that are not I/R commands. The words in the command would be converted like this:

- **tv**  
Assuming we have a device named "tv", this would match this name and the PC/IR Linc would send the command to select the "tv" bank.
- **channel**  
There is no match for this word in the "tv" bank, so it would be ignored.
- **1**  
This is the "1" key, and would send a "1" to the "tv" device.
- **2**  
This is the "2" key, and would send a "2" to the "tv" device.

You can see that you must be careful about what words you include in your voice command. If the word matches any of the names for any of the keys in the device bank you are using, it will be converted to an I/R command. For example, if your command was:

**tv mute 1 2**

This would set the device to "tv", send the I/R command for "mute", and then send the I/R commands for 1 and 2. All the words in this command match a label for a key in the "tv" bank.

Note that in the above script, the "SendIR" method has a "tv " string appended to the voice command. If you add the device you are controlling to the script, you would not have to include it in the voice command. All the following voice commands would now work on the "tv" to change the channel:

```
channel (0|1|2|3|4|5|6|7|8|9)+
go to (0|1|2|3|4|5|6|7|8|9)+
display (0|1|2|3|4|5|6|7|8|9)+
This could actually be done in a single voice command like this:
(tv channel|display|go to|channel) (0|1|2|3|4|5|6|7|8|9)+
```

---

# Chapter 7

---

## 7 RF Control Overview

HomeSeer supports the X10 MR26A RF receiver. This device receives RF commands from a variety of devices such as X10 remotes and X10 Hawkeye motion sensors. Normally, motion sensors send notification of motion over the power line using an X10 command. This presents many problems including excessive X10 collisions and long delays in controlling other devices based on motion. Using this device, the RF signal is received directly at the computer and bypasses the TM751 or RR501 RF receiver. The computer can then act on the command immediately resulting in much quicker reaction times to RF signals generated by motion sensors and RF remotes.



**MR26A RF Receiver**

### 7.1 Setting up the MR26A RF Interface

To enable the MR26A in HomeSeer, add the device as an RF interface on the interfaces tab in the HomeSeer options.

1. Open the HomeSeer options from the *View* menu, then select the *Interfaces* tab.
2. In the RF section, select the MR26A from the drop down list
3. The MR26A will now appear in the list of active devices
4. Select the MR26A in the list, then the *Setup...* button to configure the interface. From the setup dialog you can set the MR26A to relay X10 commands to other house codes and perform the same actions a TM751 or RR501 RF receiver.

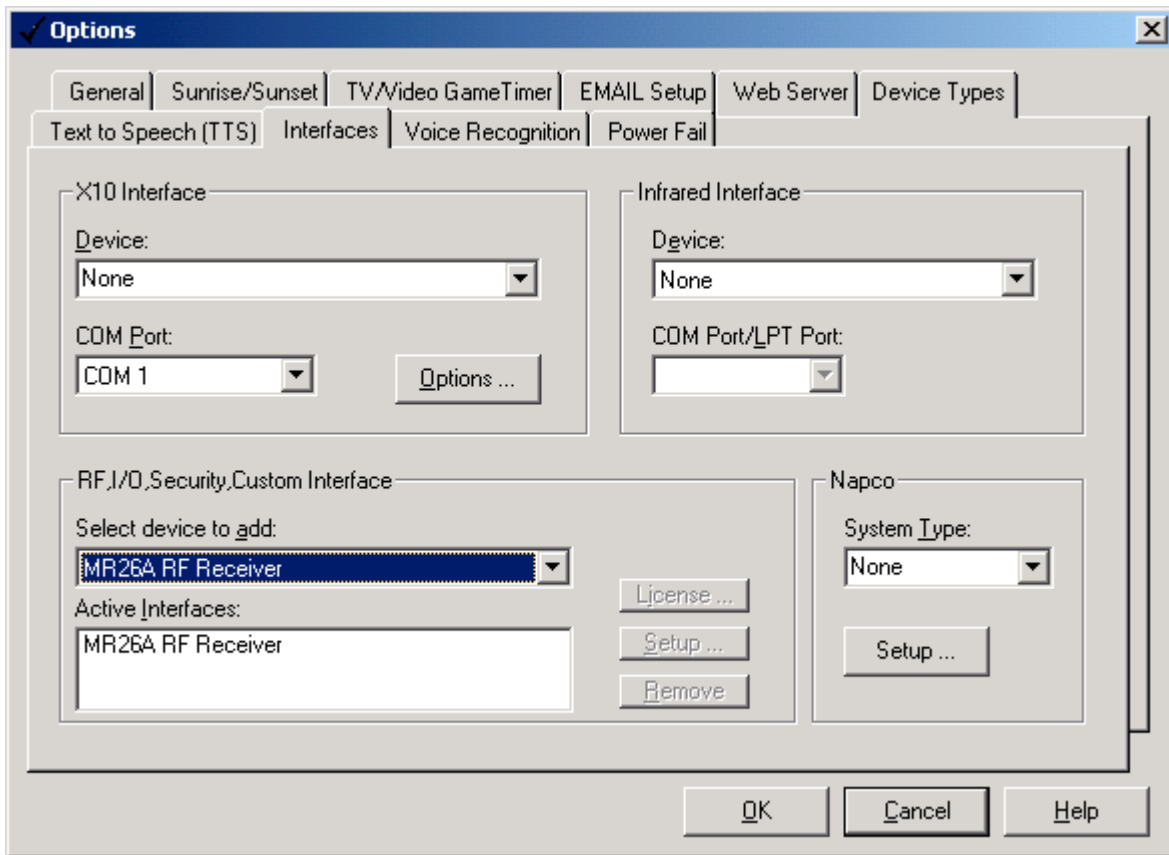


Figure 21 Setting up the MR26A

## 7.2 Setting up Events using the MR26A

When the MR26A receives RF signals, it sends the signals to HomeSeer in the form of an X10 command. This is the same command HomeSeer would receive if the RF command was relayed through an RR501 or TM751 RF receiver. To view what actual X10 command was sent, open the event log and generate motion using a Hawkeye motion sensor, or press a key on an RF handheld remote.

For more setup options, click the help button on the Setup... dialog on the interfaces tab.

---

# Chapter 8

---

## 8 Web Access

HomeSeer contains a built-in Web Server. With just a single click, you can give yourself and others access to all your X10 devices. From the web you can see the status of all your devices, as well as, add, delete, and edit all your events and devices. All the device and event configuration that you can do through the Windows interface, you can do through the web interface. In some cases, the web interface is easier to use since it uses forms for all user input. You can use either the Windows or Web interface on your local computer.

### 8.1 Starting the Web Server

To enable the Web Server, make sure you have TCP/IP installed on your computer. If you can access the Internet, then you are all set.

Bring up the options dialog from the menu View->Options... .

1. Click the web Server tab.
2. Enter the server port number. If you are not running a Web Server on your machine, then leave the default of "80", otherwise select a different port, like "81" or "8080".
3. If you would like to limit access to your HomeSeer web site, then enter a username and password. When a user attempts to access your web server, they will be prompted to enter this information. You may click the check box *Force Login* if server is idle for more than 10 minutes if you would like to force a user to login again if they stop accessing the server. Browsers will save the username and password and this would alleviate the possibility of someone gaining access to your site if a browser was left open to your page.
4. Click the check box Enable Server.
5. That's it. Now exit the dialog by clicking "OK".

The web server will start up. In the *View* pane, click on *Log*. You should see a message that tells you whether the web server started successfully or got an error.

If there was an error, check the following:

- Make sure there is no other server on the port you are using. Try changing the port to a different value, like "8080".
- Make sure TCP/IP is installed. You can test this by simply running your web browser, if that does not give you an error and you can access the Internet ok, then TCP/IP is ok.
- Send the log file to us. The log file is located in the HomeSeer application directory as "ah.log". Send it to support@homeseer.com. We will help you solve the problem.
- Try the HomeSeer Assistant Wizard. This can be displayed from the menu Tools->Assistant Wizard.... Select *View my Web Page* . Follow the prompts.



To test the server further, bring up your web browser, and enter the following URL:

```
http://localhost
```

The HomeSeer home page should be displayed.

If your computer is on the internet, you can access it with:

```
http://YOUR_COMPUTER_IP_ADDRESS:SERVER_PORT#
```

For example, if your computers IP address was 192.168.1.1 and you left the server at port 80, then enter:

```
http://192.168.1.1
```

If you set the server port to "8080" then enter:

```
http://192.168.1.1:8080
```

To get your computers IP address, check the HomeSeer log. When HomeSeer starts, it displays your IP address in the log like:

```
7/23/2002 1:32:36 PM~!~Info~!~Local IP address is: 192.168.0.14
```

Another way to get your IP address is to enter the command "ipconfig" from a DOS or Command prompt.

## 8.2 Using the HomeSeer Server with a Router

If you are using a router in your home to share multiple PC's with your internet connection, you may not be able to access your HomeSeer web server from a remote PC. To fix this problem, set up your router to forward all requests to the HomeSeer server port to the HomeSeer PC IP address. For example, if your PC IP address is 192.168.0.1 and you set up the HomeSeer web server on port 80, then set up your router to forward all requests on port 80 to IP address 192.168.0.1. Just about all routers have the ability to do this. See your routers documentation on how to set this up.

### 8.3 Device Status Web Page

The default start page for your web site displays the status of all your devices and looks like this.

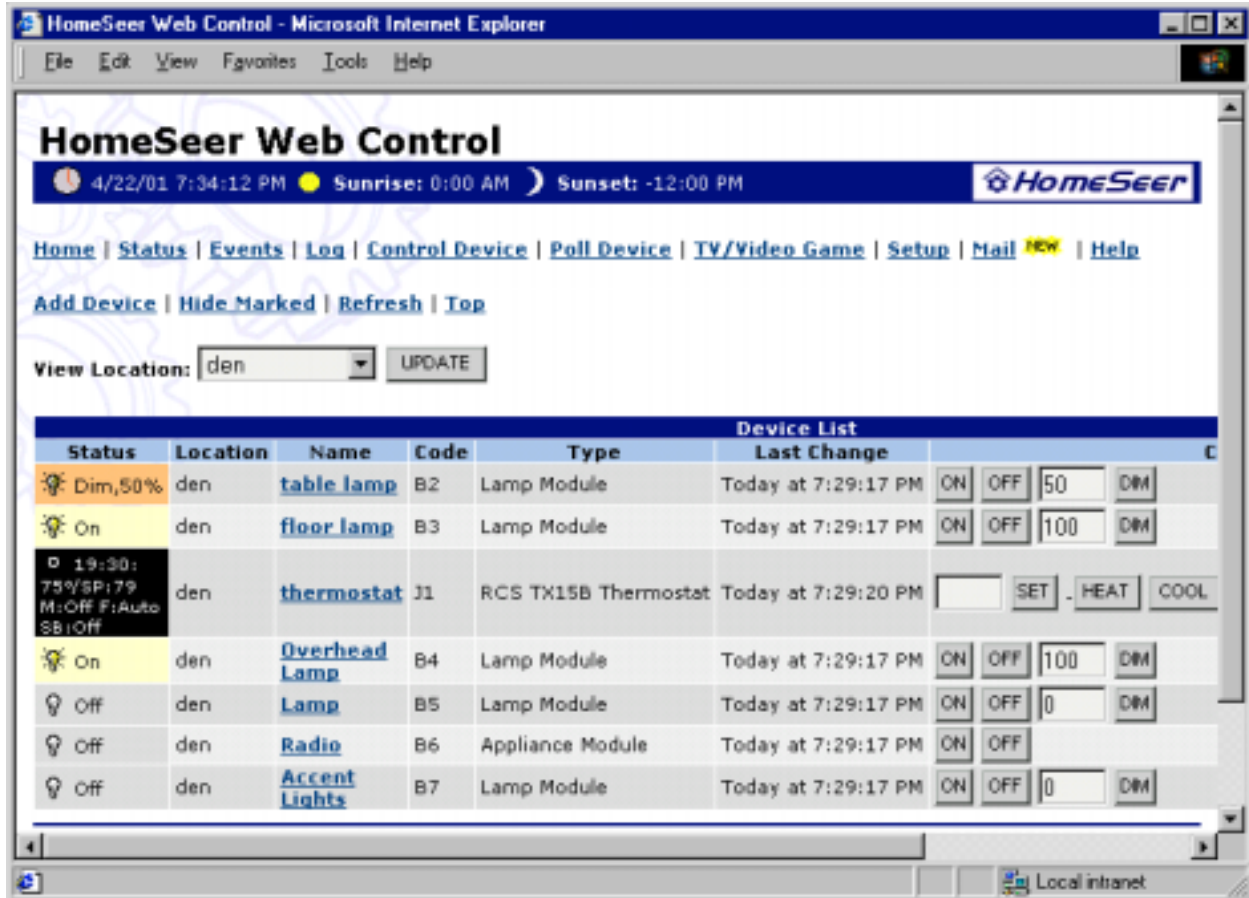


Figure 22 Device Status Web Page

There are hyperlinks along the top of the page that allow you to:

**Add Device**

Brings up the Add Device form to create a new device.

**Add Event**

Brings up a form allowing you create a new Event. The form prompts you for the event name, then brings you to the event list where you can set the event properties.

**Event Log**

Displays your event log.

**Control Panel**

Brings up a control form for directly controlling an X10 device.

**Get Device Status**

If you have any device that support the X10 Status Request command, this link will bring up a page and start polling these device in order to find out their status.

**TV/Video Game Timer**

Brings up a form to Add users or edit existing users for the game timers.

**Setup**

Brings up a form to allow you to set some configuration parameters of HomeSeer. Most configuration is done through the Windows interface, but some things can be modified from this page.

**Security**

If you have enabled a security panel interface, this link will bring up an image of the security panel keypad. Not all security panel devices support this feature. Other links may appear for different panel types.

**Hide Marked/Show All**

A device can be marked as hidden in its properties. If you click this link, all the hidden devices will be removed from the view. If the link is displayed as *Show All*, then clicking it will bring all hidden devices back into view.

**Refresh**

Clicking this link causes all devices to be updated with their latest status. The status page can be set to refresh automatically by setting a refresh time in the *Web Server* tab in the HomeSeer options.

**Log out**

If you have a username and password set for the server, this link will be available. Clicking it, will log you out of the server and you will have to re-enter your login information to display any pages.

**Top**

The web pages are *sticky* pages. This means that if you click the *ON* button for a device, the page will redisplay back at the same device. This keeps you from having to scroll back to your last location. To get back to the top of the page and remove the last *stuck* location, click this *Top* link.

**Messages**

If HomeSeer Phone is installed; this link will display a web page containing all your voice messages.

**Mail**

If HomeSeer is setup to use MAPI as your email transport, then you can read your local mail from here. This link will bring up a page that displays all of the mail that is in your inbox. Remember to keep your email client running or this page will not get updated. An icon will be displayed when new email is available. The icon file is *new\_messages.gif*, and can be replaced with a different graphic if desired.

The message list contains the following links along the top of the page:

- **Show All**  
Shows all messages that are in your inbox. Note that MAPI can be very slow if you have more than about 50 messages in your inbox. If it takes a long time to display your messages, you may want to delete some messages or move them to another folder. You can only display messages that are in your inbox. Messages in other folders, along with deleted messages, will not be available from the web interface. This is a limitation of MAPI.
- **Show Unread**  
Displays only those messages that have not been read. After reading a message, it is marked as read and will not be displayed. Note that you should use the "Back" link on the

message page so that the message list is refreshed properly.

- **Mark all as Read**  
Marks all messages as read and re-displays the message list (which will be empty).
- **Send New**  
Displays a form allowing you to compose a new email and send it.

When displaying your email messages, you can click on the subject of the message to bring up the message. After displaying the message, links along the top of page allow you to reply, reply all, forward, or delete your email message. If the message contains attachments, they will be listed as links at the bottom of the page. Clicking a link will display the attachment if it's an HTML file or prompt you to download it.

**Clicking the Delete button may permanently delete the message. This depends on the email client you are using. Some email programs simply move the message to a deleted folder.**

#### **Voice**

This link will only show up if enabled from *View->options->WebServer->Voice Web Page enabled* checkbox. This link will bring up a page that contains all of your events that are enabled for voice control. You can then use a Voice Controlled web browser to speak the links. The last link on this page is the name of your custom web page.

#### **View Location**

This drop down list may be used to filter the current device list. Select the location you wish to view, and press the *update* button. Only the devices that are in the selected location will be displayed.

## 8.4 Event List Web Page

By clicking the Event List link, a page with all of your currently defined events will be displayed. This page looks like this:

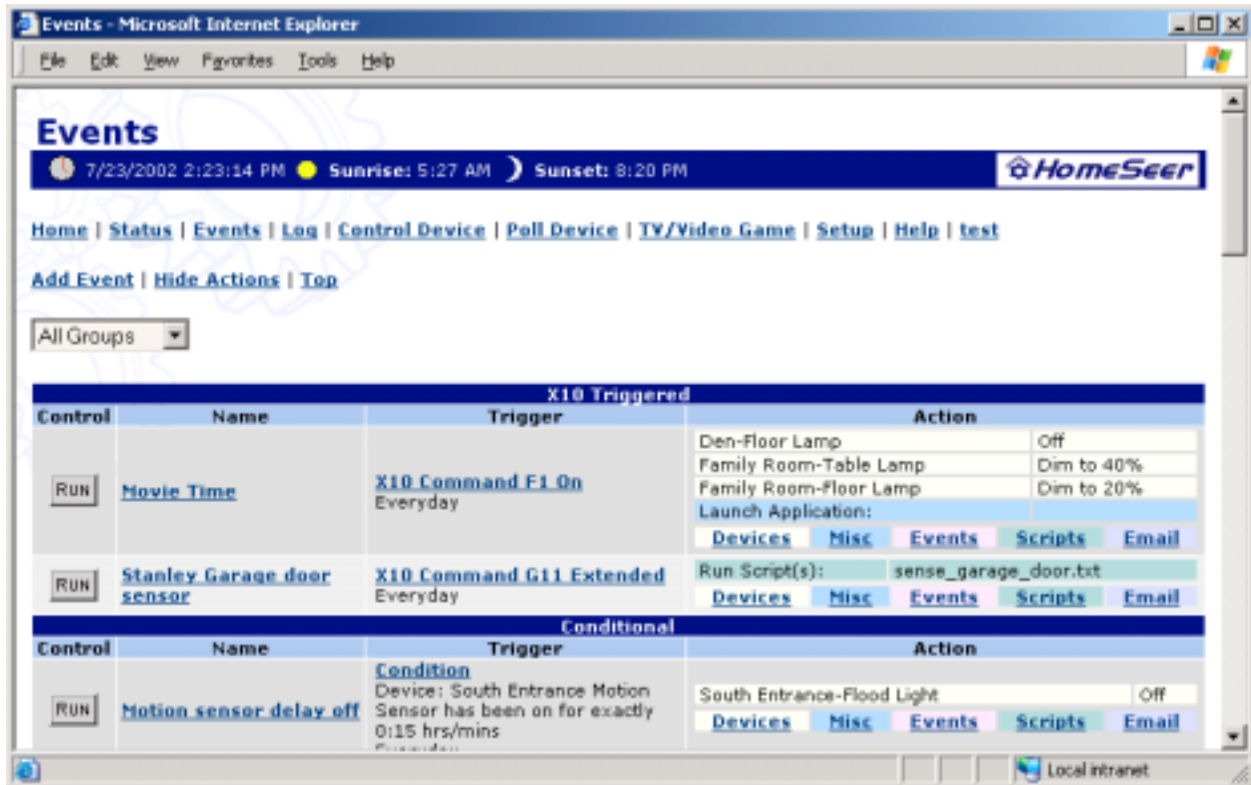


Figure 23 Event List Web Page

The links on this page allow you to customize your events. You can hide specific links from the *Web Server* tab *Customize Web Site* button in the options.

The *Hide Actions* link will hide all the actions associated with an event. This can make the page easier to read.

The name column has the following buttons and links:

- The *Run* button will force the event to trigger and execute any actions associated with the event.
- The *Event name* link brings up a form that allows you to change the name of the event, as well as disable it and mark it as a voice command. Events that are currently disabled will be displayed with a red background and events that are voice commands will have a "(V)" designation.

The trigger column has the following links:

- Clicking on the trigger brings up a page containing links for each of the trigger types. The current trigger type is in *Italics*. You can then choose the event type you want and you will be brought to a form where you can set the properties for the trigger.

The action column has the following links

- The devices link brings up a page displaying all the X10 devices this event will control.
- The events link brings up a page that displays all the events that this event triggers. Any single event can trigger a number of other events (usually referred to as macros).
- The *scripts* link brings up a page displaying all the scripts that are run when the event is triggered. From this page you can add, delete, and edit scripts.
- The I/R link brings up a page that allows you to control infrared devices. You can add and delete infrared commands.
- The email link brings up a form to enable the sending of email when the event triggers. You can set the subject of the email and the actual message.
- The security link brings up a form with the security panel actions.
- Other links may appear if any plug-ins are installed.

## 8.5 Creating your own Home Page

You can insert your own home page that will be shown before your device status page. By default, the first page a user sees is your device status. By creating your own home page and entering its name in *View->Options->Web Server->Customize Web Site->Page 1*, the page will appear as a link off of the device status page. By enabling the checkbox in *View->Options->Web Server->Customize Web Site->Page 1 displays first*, your page will be displayed before the device status page. No password or username is required to display your home page, but if you have a username and password configured, the user will need to enter it before the device status page can be displayed.

Your homepage must be located in the *html* folder in the HomeSeer application directory. There is a sample page available in the html directory named *mypage.htm* that you can use as a template. On your page you can access the device status page by adding the HTML Tag:

```
<a href="stat">HomeSeer Status</a>
```

## 8.6 Controlling devices and events from your own pages

You can add buttons to your custom web pages to control devices and trigger events. The following HTML code will run the event named "table lamp on".

```
<form method="POST">
<p><input type="submit" value="table lamp on" name="run_event"></p>
</form>
```

The following HTML code can be used to control a device directly. This code uses the name of the device rather than an event name. This code controls a device named "den table lamp". Note that the device name is combination of its location and its name.

```
<form method="POST">
<p>
<input type="hidden" name="control_device" value="den table
lamp=dim=50%">
<input type="submit" value="Lamp 50%" name="control_device">
```

```
</p>
</form>
```

Other X10 commands can be inserted other than "dim". The following commands are supported. Note that the commands are NOT case sensitive.

```
All Units Off
All Lights On
On
Off
Dim
Bright
All Lights Off
Extended
Hail Request
Hail Ack
Preset Dim
Status On
Status Off
Status Request
Dim to Off
Dimab (absolute dim, if the light is currently at 10% and the command
is light=dimab=20%, then the command bright 10% is sent)
Dimbf (bright first, the light will go to 100%, then dimmed to the
requested value)
```

To create a button that turns OFF a device named "lights" that are located in the "den", use the following HTML code.

```
<form method="POST">
<p>
<input type="hidden" name="control_device" value="den lights=off">
<input type="submit" value="Lights Off" name="control_device">
</p>
</form>
```

## 8.7 Controlling HomeSeer using ASP pages

The web server includes limited support for ASP (active server pages). Note that this is an advanced feature primarily targeted for users who are already familiar with ASP programming. All ASP pages need to reside in the HomeSeer HTML directory. Using ASP pages, HTML and script can be mixed on the same page.

Note that you cannot use HTML between the script tags "<% %>" as you can with real ASP. For example, the following ASP code will not work:

```
<%
response.write "hello"

<p>html here</p>
%>
```

You can create web pages that contain VBScript that will dynamically create the page. The web server supports the following ASP objects:

Response.write <i>string</i>	Send content back to the web browser
Response.ContentType <i>string</i>	Set the content type like text/html

The Response object supports:

- redirect
- write or send
- content type

Request.Form (.count) or ("item")

Access a form object (foreach not supported)

Request.QueryString

(foreach not supported)

Request.ServerVariables

The Request.ServerVariables object supports the following variables:

- PATH\_INFO
- REMOTE\_ADDR
- REMOTE\_HOST
- HTTP\_SERVER\_SOFTWARE
- ALL\_RAW
- HTTP\_SERVER\_PROTOCOL
- HTTP\_USER\_AGENT
- AUTH\_USER

A simple ASP page that displays some text would look like:

```
<p>Displaying standard HTML</p>
<%
response.write "hello world"
%>
```

Using response.write, a web page could dynamically create a status list of HomeSeer devices. Here is a sample ASP page that gets the status of a device and displays your PC's IP address:

```
<%
dim stat
dim ip

response.write "<b>The link is: </b>"+lnk
response.write "<p><i>this is some more stuff</i></p>"

stat=hs.devicestatus("A1")
response.write "Device A1 has a status of: "+cstr(stat)

ip=hs.Getipaddress
response.send "<p>Ip address is: "+ip+"</p>"
%>
```

Using response.form, you can retrieve information entered into a form. Here is sample ASP page that contains two buttons:

```
<%
response.write "<form action=""2_Submit_Buttons.asp"" method=""POST"">"
response.write "<input type=""Submit"" name=""action"" value=""Button One"">"
response.write "<input type=""Submit"" name=""action"" value=""Button Two"">"
```



```
response.write "</form>"
%>
```

When you click a button, HomeSeer runs the ASP page `2_Submit_Buttons.asp`. This page contains VBScript that will call `response.form` to get the submitted information. Here is the `2_Submit_Buttons.asp` page:

```
sub main(lnk)

    response.ContentType = "text/html"
    response.send "Number of pairs is:
"+cstr(request.form.count)+"<br>"
    if request.form("action")="Button One" then
        response.send "Button One was clicked"
    ElseIf request.form("action")="Button Two" then
        response.send "Button Two was clicked"
    End If
end sub
```

The proceeding scripts and pages are available in the HomeSeer HTML directory. To test them, enter the following in your web browser:

```
http://localhost/but_test.asp
```

## 8.8 View only Web Access

If you protect your web site with a password, users can log in with the *username: guest* and *password: guest*. They will be able to view your device status, but will not be able to make any changes to your devices or events and all passwords will be blanked out. This feature can be disabled from the menu *View->Options->Web Server*. Uncheck the checkbox labeled *Allow Guest Access*. By default, this is unchecked.

You can add a password to your web site from the options dialog at *View->Options->Web Server*.

You can access the HomeSeer web page quicker by embedding the username and password in the URL. For example, if your username is "bill" and your password is "tooth". Assuming your computer is at `www.bill.com`, the URL to access it would be:

```
http://bill:tooth@www.bill.com
```

You would not be prompted for a password.

If you do not wish to allow guests to see your events, uncheck the box: *Guests can view events*

## 8.9 No Password for Subnet Access

If you access your web page from inside your home, you may not want to be bothered with entering a username and password each time you access the web site. Enter the subnet of your internal network into the box *No Password needed for the following Subnets*. Any access from any of the entered subnets will not prompt the user for a password. For example, if all your internal PC's are configured with addresses like: `192.168.1.#`, then enter, `192.168`. You can also enter different subnet sizes like: `192.168.1` This will allow only addresses like `192.168.1.1` or `192.168.1.200`. `192.168.2.1` would be prompted for a password. Multiple subnets may be entered. Separate each one with a comma (,).

## 8.10 Customizing your HomeSeer Web Pages

You can change some aspects of the HomeSeer web site. From the menu *View->Options->Web Server*, click the *Customize Web Site* button.

**Figure 24 Customize Web Site**

Features customizable from this dialog are:

#### **Device Status Page Title**

This is the title that appears on the device status page

#### **Navigation Links Location**

Use the drop down list to select where the navigation links are located on the page. They can be at the top of the page, the bottom, or both.

#### **Custom Pages**

Up to four custom pages may be defined. These pages must be in your HomeSeer HTML folder. The links that are entered will appear in your navigation links on the web page. Page 1 is special in that this page will appear before the HomeSeer device status page. It is also not password protected. It can be used as a homepage for your site.

#### **Executing a function when the web site loads**

If you have a custom function on your web page, (usually embedded in the head.htm file), you can enter the function name here. This function will be executed before the page loads. This is normally a javascript function or DHTML.

#### **Adding custom HTML**

When a page is loaded, the server looks for a page named "head.htm" to load first. This allows you to encapsulate the web pages into another web page. The page "tail.htm" is loaded last. Both of these pages can be modified to contain custom HTML.

#### **Customizing the HomeSeer Website**

The web server uses CSS (cascading style sheets) to format the look of the web pages. A cascading style sheet is a text file that describes the fonts and colors used on the page. Click the Edit Stylesheet button on the Customize Web Site dialog in the options. The style sheet

contains tags that represent different areas of the web page. The tags are:

<b>Tag</b>	<b>Page item it affects</b>
body	page background color and fonts used on the page
A:link	default link text and font
A:visited	link color and font for visited links
A:hover	link color and font when the mouse pointer is over a link
.activetrigger	the asterisk next to the active trigger when displaying the actions for an event
.formbutton	appearance of all buttons (size, font, and color)
.formtext	appearance of text in form text boxes
.logobar	appearance of the bar containing the time and HomeSeer logo
.pagetitle	format of the page title at the top of each page
.tablecellapp	application table cell in event view (action table)
.tablecellcustom	table cell in device list when a custom string is assigned to a device status
.tablecelldisabled	event list name cell appearance when the event is disabled
.tablecellgreeting	phone action cell in event view (action table)
.tablecelloff .tablecellon .tablecelldim .tablecellunknown	table cell in device list, device status. Depending on the status of the device, the appropriate tag is applied to the table cell
.tablecellphone .tablecellsecurity .tablecellscript .tablecellspeak .tablecellemail .tablecellir .tablecellmisc .tablecellevent .tablecelldevice	Link cells in action table in event view
.tablecolumn	column header in event view
.tableheader	header column in event view
.tablerowevent .tablerowodd	Table cell colors for cell backgrounds for even and odd table rows

## 8.11 Remote File System Access

You can access the file system on a remote machine. If you know the directory and filename of a file on the remote machine, you can download it with the following link. Let's assume you wanted to get the file "ah.log" in the HomeSeer directory. The link would be:

```
http://192.168.1.1/./ah.log
```

You can back up and down other directories. If HomeSeer was installed in c:\program files\homeseer, and you wanted the file in c:\temp\test.doc, you would enter

```
http://192.168.1.1/../../../../temp/test.doc
```

If you are not logged into the remote machine, you will not have access to any of the files on that machine.

This feature can be disabled from the web server tab in the options. It is disabled by default.



---

# Chapter 9

---

## 9 Security Panel Support

### 9.1 Setting up the Napco interface

HomeSeer includes support for interacting with security systems. Support for the Napco Gemini system is included in the 3<sup>rd</sup> party folder on the CD. The Napco panel communicates with HomeSeer through an available serial port. The system can notify HomeSeer about any event that occurs in the system. This includes sensor status change, and alarm events. You can then trigger events based on these responses.

Note that a third party supplies the Napco support. The HomeSeer distribution contains a free 30-day trial of the HAWizard Napco control in the 3<sup>rd</sup> party folder. This control provides the interface between HomeSeer and the security panel. You will need to purchase the control for a nominal fee to continue using it in your system. More information about this control can be found at [www.hawizard.com](http://www.hawizard.com).

In order for your Napco system to communicate with HomeSeer, you will need to upgrade your security panel with the Napco HA upgrade. The part number for the upgrade is NA-GEMAUTOMA. It's available at [www.basshome.com](http://www.basshome.com).

To install the Napco support:

1. Install the HAWizard software by running the HAWizard Napco installer that is located in the 3<sup>rd</sup> party folder
2. Copy the file `hssec.ocx` located in the 3<sup>rd</sup> party folder to your HomeSeer folder.

The next time you run HomeSeer, go to the interfaces tab in the options. Under Security, select Napco from the drop down list. Click the Setup. button. Enter the master code for your system and COM port your system is connected to. (remember to enter the master code for the HA upgrade and not your original master code) From here you can also select the zones you would like to bypass.

**When selecting zones to bypass, only select the zones that are configured in your panel. Unpredictable results will occur if you select non-configured zones.**

Click OK. If HomeSeer was unable to connect to the panel, an error box will be displayed.

You may have to restart the application after setting your security code the first time.

To test that the system is functioning properly, display your HomeSeer web page by clicking the globe icon on the toolbar. Click the Security link. A new page will be displayed with an image of your Gemini keypad. If the *Not Connected* message is displayed, try restarting HomeSeer and check it again. Also check your event log for any error messages.

To set triggers based on panel events, see the security panel trigger section.



---

# Chapter 10

---

## 10 HVAC Support

### 10.1 Setting up thermostats

HomeSeer supports HVAC devices through the use of scripts. The following script is used to communicate with the RCS TX15B thermostat:

```
rsc_tx15b.thm
```

This script supports the *RCS TX15B bi-directional X10 thermostat*. Other scripts are available that support the RCS TR15 and HAI RC80.

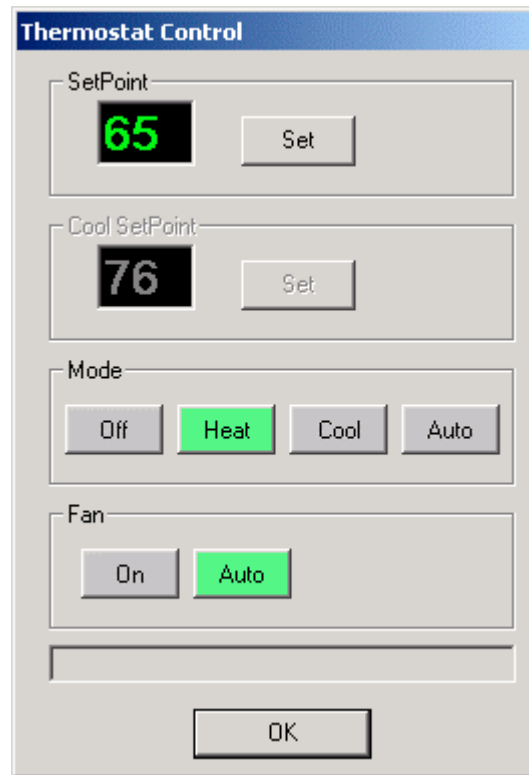
Thermostat scripts all have the extension ".thm". Check the HomeSeer updater from the help menu for updates for other thermostats.

To create a device that will read and set a thermostat, do this:

- Ensure that you have a thermostat device type created. You will need this device type when you create your thermostat device.
  - a) Go to the HomeSeer options and click the *device types* tab
  - b) Look in the list at the left and find the device type named *RCS TX15B Thermostat*
  - c) Click this device and make sure the *Thermostat* check box is selected, and the proper script appears under it
  - d) If the Thermostat checkbox is not selected, select it, and select the proper script
  - e) If no thermostat is defined in the list, enter a new name for the thermostat in the *Enter new type* box, click the *Thermostat* checkbox, and select the script. Then click the *Add It* button to save the new device type.
- 1. Close the option dialog and click the *Add Device* button on the tool bar. A *new device* dialog will appear.
- 2. Enter a name for the device like *Hallway thermostat*, and a location like *downstairs*
- 3. In the device type box, select the thermostat you just created in step 1. The name by default is *RCS TX15B Thermostat*.
- 4. Select the base address for your thermostat. The RCS thermostats use an entire housecode. If you have your thermostat set to use housecode J, then set the device code to J1.
- 5. Click OK. Your thermostat is now configured.

To test communication with your thermostat, right click the thermostat device in your device view. The pop-up menu will have a single option: *control...* Select this, and a thermostat control dialog will appear.





**Figure 25 Manual Thermostat Control**

It will take a few seconds for HomeSeer to poll your thermostat and get its current settings. If communication was successful, your current settings will be green. All the buttons will be gray if communications failed. You will also see the status of your thermostat in the device view.

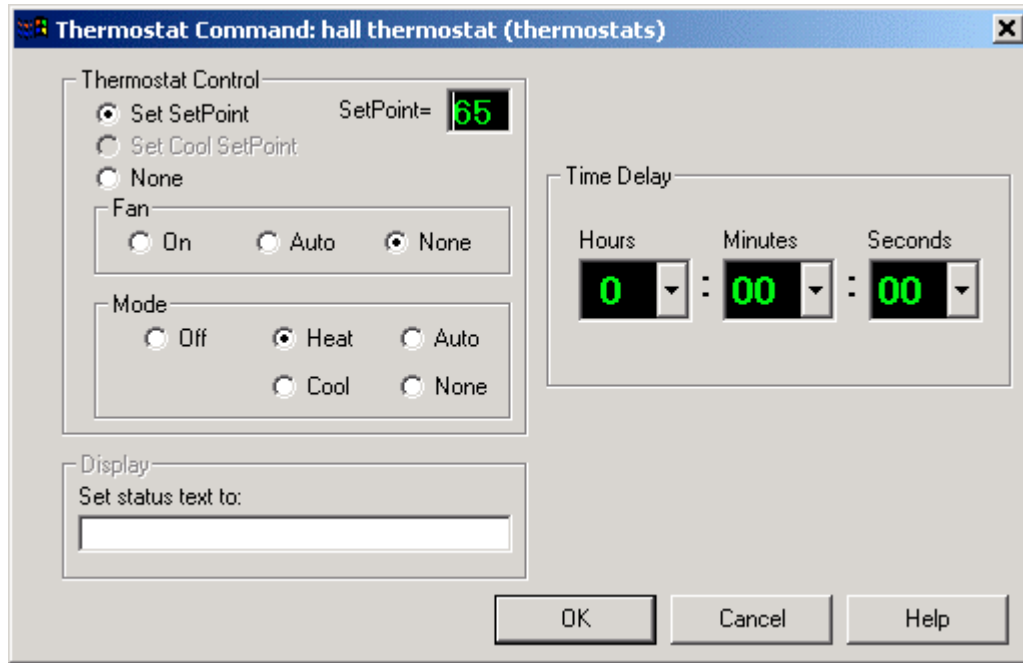
You can now change the current settings of your thermostat.

## 10.2 Controlling a thermostat with events

An event can be used to control your thermostat also.

To control a thermostat from an event, do this:

1. Create a new event, set the desired trigger
2. Click the *Device Actions* tab
3. Click your thermostat device in the upper list and drag it to the lower list. This adds the device to the event
4. Right click, or double click the device in the lower list. A *thermostat control* dialog will appear.



**Figure 26 Thermostat Control Dialog**

5. Select the actions you wish performed on the thermostat when the event triggers. Make sure the Fan and Mode are set to *None* if you do not wish to change these settings.

### 10.3 Creating new scripts for other thermostats

If you have a different thermostat that you would like to control, you can create your own script for it. Copy the `rsc_tx15b.thm` thermostat to a new name like `superstat.thm`. Now modify all the functions to communicate with your thermostat. The scripting language supports the RS232 ports on the PC so you can control RS232 thermostats also.



---

# Chapter 11

---

## 11 Power Failure support

HomeSeer has the ability to recover from a power failure and restore all your devices to their proper state. When power is restored, HomeSeer can go back and re-run all events that should have run while the power was out. The actions to the events are not actually run, but the state of the controlled devices is saved. After all events are run, the devices are then updated by sending the proper commands to them. This includes thermostat devices, so heating and cooling system set points can be set properly.

### 11.1 Enabling Power Failure Support

To enable the power fail support, check the box *Enable event catch-up on power restore* on the *Power Fail* tab in the options. Select the number of hours you want HomeSeer to go back when synchronizing events. For example, let's suppose that you lost power during a storm at 11:0am. Let's further imagine that you have an event that is triggered at 1:00 PM, but the power is not restored until 8:00 PM. You would want HomeSeer to look back at least 7 hours so that event was included when determining the status of any controlled devices. If that event turned on a light, the light would be restored when the power came back on. When power is restored, HomeSeer resets its internal clock back the number of hours you entered in the *Power Fail* configuration, then executes each event while incrementing time. This way, it will run all missed events and update the status of all your devices. Certain actions are not executed while the catch-up is taking place. Any action that does not affect the status of a device, such as speaking, is not run.

Optionally, scripts can be set to not run. If you have many events that run scripts, it could take quite a while for HomeSeer to run through all the missed events. Normally you would want this checkbox unchecked.

### 11.2 Disabling Power Failure Support on an Event/Device basis

You may have some devices that you do not want included in the power fail recovery. For these devices, uncheck the box *Include in Power Fail Recovery* in the properties for these devices. These devices will not be controlled when power is restored.

If there are any events that you do not want included in the power fail recovery, uncheck the box *Include in Power Fail Recovery* in the properties for these events. These events will not be executed when power is restored and all events are run.



---

# Chapter 12

---

## 12 Printing

### 12.1 Printing event lists and controller labels

From the *File->Print* dialog you can print the following:

- A listing of all your devices
- A listing of all your events
- Controller labels for 8 button handheld controllers, 4 button tabletop controllers, and 4 button wall controllers
- If you have an unsupported controller, you can create custom labels just by entering some information about the size of the labels

When printing your devices, the sort order of the print out is dependent on the current devices view. Sort your devices the way you want them printed.

When printing labels, you must enter the housecode of the devices you want to print. For example, entering housecode A , will print the device names of all devices in that housecode (up to the number of buttons on the specific controller).

When printing labels for your devices, if a device does not exist for a specific house code-unit code, the program will search your event list for an event that has its trigger set to "X10 Command". If one is found, this is used for the label. This is handy if you have an event name "Movie" that is triggered by a tabletop keypad button "A1". You really want the label for the event to be printed rather than the name of the device (which may be labeled "keypad A1").

**Note that if you intend to print labels for your devices, keep the device names to 10 characters or less. Otherwise the labels will not print correctly.**



---

# Chapter 13

---

## 13 TV/Video Game Timer

### 13.1 Setting up the TV/Video Game timer

HomeSeer has the ability to track the use of any device connected to an X10 controller. If you would like limit access to the television or a video game console, you can create accounts for family members and give them a fixed amount of time to use the device. For example, you can give Bill 1 hour a day to play video games and give Sue 6 hours a week to watch TV. When the user uses up his/here time, HomeSeer will turn off the device. It is very inexpensive to setup and uses a single X10 appliance controller about (\$8.00) and a 4-button tabletop controller about (\$11.00). This is how you set it up:

- 1 Enter the configuration dialog through the menu *View->Options->TV/Video Game Timer*.
- 2 Under *Select User*, it displays *Create New User*, if not, click the drop down list and select *Create New User*
- 3 Enter the users name under *User Name*.
- 4 If you are going to limit access daily, click the radio button next to *Daily Max Time* and select the number of hours and minutes you would like to limit daily access to. Daily access is reset to this number at midnight every day.
- 5 If you are going to limit access on a weekly basis, then click the radio button next to *Weekly Max Time* and enter the number of hours and minutes you would like to limit weekly access to. Weekly access time is reset every Sunday night at midnight.
- 6 If you would like to temporarily give this user unlimited time, you can check the check box labeled *No Time Limit*. The user will have no limit as long as this is checked.
- 7 You can view remaining time for the user here also.
- 8 Now set the password the user must to enter to access the device (TV or Game). The password is entered into the tabletop controller. Note that the controller only has 4 keys, for codes 1-4. Codes 5-8 can be accessed by sliding a switch, but it's easier to limit the password to a combination of the first 4 digits, like "2-1-3-4". Enter the password under *Password*.
- 9 The tabletop keypad must be set to a house code that is not being used by any other devices in your home. If you set it to A and you have device modules at A, then anytime someone enters a password, your lights would come on. Let's assume you have no devices on house code P, set the tabletop controller to P, and enter P under *Password Housecode* in the dialog. HomeSeer will now listen for the password on this housecode.
- 10 Now set the *housecode* and *device code* of the appliance module that the tv or video game is plugged into. This can be any code, but it's best to set it to the house code you picked for the tabletop controller and device code 16. For example, if you chose house code P for the tabletop controller, set the appliance module to P16 and under *Appliance Module Address* enter P for the house code and 16 for the device code.. Since the table top controller can only access codes P1->P8, there is no way your child can bypass HomeSeer and enable the TV. (He could always unplug the module, but then they you'd ground him for a month!)



- 11 To keep prying eyes from modifying this configuration, you can enter a password under *Tab Password*. This will cause HomeSeer to prompt you for a password whenever someone tries to access the *TV/Video Game Timer* tab in the Options dialog.

### **Using the TV/Game Timer**

To enable the TV or any device setup for timed access, the user must first press any button on the tabletop controller to OFF. This tells HomeSeer that a password is ready to be sent and resets its password counters.

The user then enters his/her password using the buttons on the controller. He/she must press the buttons **ON**. If the password is correct, HomeSeer will enable the device and start timing how long it is ON for that user.

You can track device usage by looking at the event log and you can view remaining time on the *TV/Video Game Timer* dialog, or via the web interface. The web interface will display remaining times without a password if the user logs in with the guest *username* and *password*.

# Chapter 14

## 14 Options Summary

### 14.1 Options General

Option	Description
Enable event logging to file "ah.log"	Events are logged to a file as they are detected. Note that this file is never deleted and may become very large. You can delete it whenever you like. A new file will be created automatically.
Enable logging to event view	When checked, events are logged in memory and displayed in the event view window.
Max size for event log (memory only)	Allows you to limit the amount of memory that the event log will use. When the log fills, oldest events are removed from the log as new ones arrive.
Always minimize to system tray	If enabled, the application will always minimize to the system tray when you click the minimize button.
HomeSeer Phone (launch at startup)	If HomeSeer Phone is installed, checking this box will allow HomeSeer to launch HomeSeer Phone when it starts up.
Dim increment 5% for all slide controls	When using the dim slider in dialogs that allow you to dim a device, the slide control will snap to 5% increments making adjustments easier.
No prompt startup	When checked, the program will start up without any dialogs, even if there was an error. This is useful if the application is being run at a remote location where no one would be available to acknowledge a prompt. Errors are logged to the event log.
Close box minimizes application	If enabled, HomeSeer will minimize to the system tray when you click the applications close box.
Startup in system tray	The application starts minimized to the system tray. Right click the icon in the system tray to restore the application.
Scripts cannot timeout	Normally, a script will timeout after 30 seconds if it does not call <code>hs.WaitSecs</code> or <code>hs.WaitEvents</code> . If you call a time consuming function like <code>hs.GetURL</code> , you may not have control over the amount of time the function takes. Selecting this option will remove the

	script timeout. Use with caution. If a script does "hang" you will not be able to stop it.
Security Offset	If you have the security checkbox selected in an event that is triggered by time, the actual trigger time is skewed by a random amount. The option sets the maximum and minimum time in minutes that the event trigger will be altered. For example, if the setting was 30 minutes, and the event was set to trigger at 7:00 PM, then the trigger could be a max of 7:30 PM or a min of 6:30 PM.
Restore default settings	Restores all program defaults. You must restart the application for the changes to take affect.

## 14.2 Options Sunrise/Sunset

Option	Description
Longitude/Latitude	Enter the longitude and latitude for your location. Longitude values should be positive for locations east of longitude 0 and negative for locations west of longitude 0. Press the calculate button after entering your values.
Pick Location	Select a location that is closest to you. The list currently only contains US locations.
Calculate	Press this button after you have entered your own longitude and latitude values.

### 14.3 Options EMAIL Setup

Option	Description
Default Subject	If you are sending email notifications, this is the text that will appear in the subject field of the message
Default Message	If you are sending email notifications, this message body text will appear in the message.
Check for mail every # minutes	<p>If checked, will check for new email every number of minutes entered. If you are using POP for your email, the program will check your POP email account.</p> <p>If you are using MAPI for your email, it will check your email client for new email. Your email client must be setup to check for mail with your real email account.</p>
Use MAPI to handle email	<p>Preferred method for checking email. The program will use the MAPI email interface to send and receive email. You must be running an email client that supports MAPI. Clients such as Outlook Express and Outlook98 support this. Make sure MAPI is enabled in the email client. If it is not, the program may hang when MAPI is enabled.</p> <p>If the program does hang on startup, you can restore the factory defaults by pressing the "restore defaults" button.</p> <p>If you are using an exchange server to handle your email, set your profile name and password and HomeSeer will log you into the server when it starts up. This prevents the login dialog from popping up.</p> <p>If this is not checked, the program will attempt to use POP to handle email. If you do not want to use email at all, leave the "SMTP Server" field blank, and uncheck the "check for email" checkbox.</p>
EMAIL Notifications are sent to this address	If an event is set to send email as a notification, you must enter the notification address here.
SMTP Server/POP Server	If you are using POP to handle email, enter information about your SMTP and POP servers here. This information is available from your ISP and is the same information you entered into your email client.

## 14.4 Options Web Server

Option	Description
Server Port	Sets the port the server runs on. Can be any value from 1 to 9999. Standard web servers run on port 80. You can "hide" your site by picking a different port.
Enable Server	When checked, the web server will attempt to run. Check the event log for startup errors.
Auto refresh every # seconds	Enter the number of seconds between page refreshes. This will force the browser viewing your status page to refresh itself. Set the value to 0 to not cause a refresh.
Log errors	Select this option to log all web server errors to the event log. Logging this information can clutter up the event log. Uncheck it to avoid logging this information.
Log Logins	Logs the IP address of the host that logged into the web server.
Web Username/Password	You can password protect your site by entering a username and password here.
Force login if server is idle for more than 10 minutes	This will force users to login if the server has not seen any activity for more than 10 minutes.
Allow guest access	People can login to your site as guest with the username "guest" and the password "guest". As a guest, they can view your devices and events, but cannot make any changes, or control any devices. If unchecked, no guest access is allowed.
Guest can view events	Check this if you would like guests to view your events web page. You may not want guests to see this information, if so, uncheck this option.
No password needed for the following subnets	Enter the subnets that are local and private. When a host from one of these subnets accesses the server, they will not be prompted for a username or password. Enter subnets like:  192.168, 192.168.1, 172.29, 172  Separate multiple subnets with a ",".
Server Stats	Displays the number of page views on your server.
Voice Web page-enabled	When checked, a special web page is available on your site that can be navigated with a voice controlled web browser.
Voice Web page-display all devices	When checked, all your devices will be listed on the voice control web page.
Appearance	Displays a dialog that allows you to customize the

	colors on your web pages.
Only allow posting to server	If your ISP does not allow servers on their network, you can still control devices in your home. By checking this option you can post data to the HomeSeer web server, but GET requests will be ignored. Your server will not be visible on the Internet.
Allow access to local file system with ../ from web	Enables the ability to access your file system from the web. Use with caution.

## 14.5 Options Voice Recognition

Option	Description
Enabled	Check this to enable voice recognition. You may have to restart HomeSeer before this setting will take affect.
Recognition Engine	If you have installed another recognition engine, you can select it here.
Speak Recognized Commands	The computer will echo commands as they are understood
All Commands Require Confirmation	The computer will ask you if you really want it execute the given command. Answer "yes" or "no". This affects all commands. To selectively enable confirmation on a voice command basis, select the "confirm" check box in the properties for the event.
Attention Voice Command	The computer must hear this phrase before it will start listening for commands. You should not make this a simple phrase as the computer may hear it in background conversations.
Attention Acknowledge Phrase	The computer will speak this phrase when it hears the attention phrase.
Ignore Voice Command	If a phrase is entered here, this phrase must be spoken to stop the computer from listening for commands. Once heard, the computer will start listening only for the attention phrase. This feature is useful if you want to give the computer a bunch of commands and do not want to keep saying the attention phrase before each one.
Ignore Acknowledge Phrase	When the computer hears the ignore phrase, it will speak this phrase.

## 14.6 Options Text to Speech (TTS)

Option	Description
Enable text to speech	When checked, text to speech is enabled. The type of speech is determined by the "Use MSAGENT character to speak" setting.

Voice	If you have multiple voices installed on your computer, you can select the voice that the computer uses to speak.
Use MSAgent character to speak	If checked, the agent character will speak any desired text. If unchecked, the computer will speak without the character. You cannot use the Agent character with SAPI5 voices. These are voices with "(V5)" in their name.
Hide agent after speaking	The agent character will disappear and reappear when necessary.
Character name	Select the desired character from the list. This character will replace the current one.
Log Speech Phrases to Event log	If checked, all spoken phrases will be logged to the event log.

### 14.7 Options Power Fail

Option	Description
Enable event catch-up on power restore	When checked, HomeSeer will attempt to update the status of your devices when power is restored. Device status is determined by going back in time the number of hours configured and running all missed events. Commands are then sent to all devices configured to participate in the recovery process.
Number of hours to synchronize from	The number of hours HomeSeer will go back and start running events from.
Do not run scripts while updating	Check this to not run any scripts while catching up during a restore process. Running scripts can severely slow the recovery process.

---

# Chapter 15

---

## 15 Scripting

One of the more powerful features of HomeSeer is the ability to create scripts that allow you to add an unlimited amount of "if- then-else" logic to your events. The scripting engine is provided by Microsoft and supports VBScript by default. You can use any scripting engine that is compatible with MSScript. HomeSeer includes the VBScript language. If you have Microsoft Internet Explorer installed, then you already have VBScript and Javascript installed. See the section on Script Language for more information on how to change the script language that HomeSeer uses.

### 15.1 Creating a Script

All scripts must reside in the scripts folder in the HomeSeer application directory. Scripts have the extension ".txt" or ".vbs" and are simple text files. You can edit your scripts in this directory or from the *Event Properties* dialog. Here are the steps to create a simple script that turns a light off if it's on:

- In the *Events View*, click on the *add event* button. A new *Unnamed* event is created.
- Right click this event and open the *Event Properties* dialog, give it a name like *light test*.
- Now click the *Scripting/Speech* tab.
- Click the *New...* button. A dialog box displays asking you for the name of the script. Enter *lightoff*, click Ok. Note that for VBScript scripts, the extension for the script is ".txt" or leave it blank and ".txt" will be added. For JScript enter the extension as ".js" and for Perlscript, enter the extension as ".pl".
- The new script will be displayed in whatever text editor you have configured to handle ".txt" files. It's usually *Notepad*.
- The script already has the main subroutine defined for you. Modify the script so it looks like this:

```
sub main()  
  if hs.ison("B2") then  
    hs.execx10 "B2", "off", 0  
  end if  
end sub
```

- Save the script.
- To test it, click on the name of the script in the list box, and then click *Add Selection*. This adds the name of the script to the run box. Now click the *test* button. The light at code "B2" will go off if it's on.

### 15.2 Executing single script statements

In the Run Script(s) text box you can also 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 inserted into the *run box*:

```
&if hs.ison("b2") then hs.execx10 "b3","off",0 else hs.execx10 "b3","on",0  
  
or  
  
&hs.SetDeviceString("b2","Garage Open")
```

**Multiple statements may be added to the run script box. Separate each statement with a colon (:).**

**Note that only "hs" is supported as an object for script statements and not "ah".**

### 15.3 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.

### 15.4 User supported scripts

The HomeSeer message board has a section for scripts that are donated by users. There are a large variety of scripts available. Please visit the message board at:

<http://www.homeseer.com>

Click the message board link at the top of the page.

Script packages are also available from the HomeSeer updater. Select *Updates* from the Help menu in HomeSeer for a list of available packages.

#### Script Language

VBScript contains a large number of built in commands. If you are new to scripting, get the vbscript documentation from Microsoft. A tutorial on VBScript is also available. You can get them off the internet at:

<http://msdn.microsoft.com/scripting>

If you would rather use JScript instead of VBScript (or any other supported script language) you can change the language HomeSeer uses by using a different extension with the script filename. The following scripting engines are supported:

- .txt and .vbs files are VBScript
- .js files are JavaScript

- .pl files are PerlScript

## 15.5 Common scripting questions

Here are some answers to some common scripting questions.

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

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:

```
hs.ExecX10 "A1", "on"
```

The following function returns a status value:

```
if hs.IsOn("A1") then
```

- *Are the function names case sensitive?*

No. The function `hs.IsOn("A1")` and `hs.ison("A1")` are identical and work the same way.

## 15.6 Writing and reading from files using VBScript

When HomeSeer is installed, a DLL is installed in your system folder that adds file system support to VBScript. There is a possible security risk in loading this DLL that you should be aware of. Since Internet Explorer uses VBScript, it is possible for a web page to contain a script that could read and write your file system. Note that if your security settings are set at their default settings, Internet Explorer will NOT allow any script to run that references the file system functions in the `scrrun` DLL. If you are concerned about this, simply delete the file `scrrun.dll` from your system folder. More information, as well as sample code, can be found at Microsoft's web site at [msdn.microsoft.com/scripting](http://msdn.microsoft.com/scripting). Refer to the section on the "File System Object".

## 15.7 HomeSeer Scripting Interface (function quick reference)

Along with the built in VBScript commands, you can control many aspects of HomeSeer. There are commands available to control X10 devices by housecode/unitcode and by the name you have assigned to them. You can force the trigger of events and cause the MSAgent to speak. Here is a description of available commands. All scripts must reside in the scripts folder in the HomeSeer application directory. There are some sample scripts in the folder to get you started. Note that all these functions have to be preceded with either "hs." or "ah." For example, to write a message to the event log, the function WriteLog is used like this:

```
hs.WriteLog "error", "something bad happened"
```

### Misc Functions

<i>Function Name</i>	<i>Operation</i>
ClearLog	Clears the event log
CreateVar	Create a global variable by name
DeleteVar	Delete a global variable
ftp	ftp a file
FTPLastError	Return last FTP error number
ftpSetProxy	Set proxy server settings for FTP access
SetRemoteTimeout	Set timeout for Internet access
GetURL	Return the text of web page
GetURLIE	Use Internet Explorer to retrieve a URL
GetAppPath	Returns a path to the HomeSeer executable file
GetIPAddress	Returns IP addresses assigned to the system
GetPlugins	Get all installed plug-ins
IsScriptRunning	Check if a script is running
plugin	Access functions in a plug-in
GetVar	Retrieve a variable
GetPrinter	Get printer object and set properties
PrintOut	Print text to a printer
GetINISection	Get an entire section from an INI file
GetINISetting	Set a single setting from an INI file
ClearINISection	Clear an entire INI section
SaveINISetting	Save a single setting to an INI file
Keys	Send keystrokes to a running application

---

Launch	Launch an application
MuteSpeech	Silence all speech output
NoLog	
Ping	Check a remote host is alive
PlayWavFile	Play a WAV file
PlayWavFileEx	Play a WAV file out a specific audio device
RegisterStatusChangeCB	Register a callback script for device status changes
Run	Run another script
RunEx	Run a function in a script with a parameter
ScheduleFile	Set or get the current configuration file
StringItem	Return sub strings
Sunrise	Return the time of sunrise
Sunset	Return the time of sunset
SaveVar	Save a variable
SetVolume	Sets the volume of the WAVE device
Shutdown	Shutdown the HomeSeer application
System	Access the system object
SystemUptime	Return the uptime of HomeSeer
GetVolume	Get the current volume from an audio device
SetSecurityMode	Enable/Disable variable timing for events
SystemUptime	Return the time HomeSeer has been running
version	Return the version number of HomeSeer
WaitSecs	Waits the given number of seconds
WriteLog	Write a text message to the event log
WaitEvents	Allow other processes to run
WEBStatsPageViews	Get/Set the web site page view statistics
WEBLoggedInUser	Get last user logged into the web server
WEBValidateUser	Validate a username/password
X10InterfaceStatus	Returns the status of the configured X10 interface

---

**Media Functions**

<i>Function Name</i>	<i>Operation</i>
MediaMute	Mute a selection
MediaPlay	Play a selection as set with MediaFilename
MediaVolume	Set the volume of a selection
MediaPause	Pause the playing of a selection
MEDIAFilename	Set a filename to play
MedialsPlaying	Check if a selection is playing
MediaStop	Stop playing a selection

**Communications Port Functions**

<i>Function Name</i>	<i>Operation</i>
CloseComPort	Close a COM port
GetComPortCount	Get the number of characters waiting at a COM port
GetComPortData	Get the data from a COM port
OpenComPort	Open a COM port
SendToComPort	Send some data out a COM port
SetComPortRTSDTR	Set com port DTR and RTS signals

**Infrared Functions**

<i>Function</i>	<i>Operation</i>
SendIR	Send infrared command

**Voice Related**

<i>Function Name</i>	<i>Operation</i>
AddVoiceCommand	Adds a voice command to be recognized
ClearAllVoiceCommands	Removes all voice commands added with AddVoiceCommand
LastVoiceCommand	Returns the string of the last recognized voice command

LastCommandSelected	Returns the name of the event for the last voice command
ListenForCommands	Toggle listen for commands for attention phrase only
MuteSpeech	Do not speak
StartListen	Start listening
StopListen	Stop listening
StopSpeaking	Stop speaking current phrase
Speak	Speak some text
SpeakEx	Speaks text out a specific audio device
SetSpeaker	Set the current voice recognition speaker
ListenMode	Get current recognition mode
SpeakToFile	Save text-to-speech to a file

### Device Functions

<i>Function</i>	<i>Operation</i>
ClearLastX10	Clear out information about the last receive X10 command
DeviceButtonAdd	Add a button to a device
DeviceButtonRemove	Remove a button from a device
DeviceCount	Returns the total number of devices configured
DeviceExists	Check if a device exists
DeviceLastChange	Get the last change time for a device
DeviceValue	Returns the value associated with a device
DeviceValueByName	Returns the value associated with a named device
DeviceStatus	Returns the status of a device as an integer
DeviceString	Returns the text string associated with a device
DeviceStringByName	Returns the text string associated with a named device
DeviceTime	Returns the time in seconds since last status change
DeviceTimeByName	Returns the time in seconds since last status change

---

ExecX10ByName	Send X10 to named device
ExecX10	Send X10 to device
GetDevice	Returns a reference to a device
GetDeviceByRef	Get a device using its reference number
GetDeviceEx	Get device by name
GetDeviceCode	Returns the house code unit code for a named device
GetDeviceList	Get all devices
IsOnByName	Returns TRUE if named device is ON
IsOffByName	Returns TRUE if named device is OFF
IsOff	Returns TRUE if device is OFF
IsOn	Returns TRUE if device is ON
LastX10	Returns information about the last X10 command received
NewDevice	Creates a new X10 device
NewDeviceEx	Creates a new device by name
PollDevice	Polls a device for its status
RegisterStatusChangeCB	Register a script for device status changes
UnRegisterStatusChangeCB	Unregister a status change callback
SetDeviceLastChange	Set last change time for a device
SetDeviceStatus	Set the status of a device
SetDeviceString	Sets the status of a device to a text string
SetDeviceStringByName	Sets the status of a named device to a text string
SetDeviceTime	Set the elapsed timer for a device
SetDeviceValue	Set the value of a device
SetDeviceValueByName	Set the value of a named device

---

**Event Functions**

<i>Function</i>	<i>Operation</i>
DelayTrigger	Delays execution of an event by the number of seconds
DeleteEvent	Deletes the named event
DisableEvent	Disables the named event
EnableEvent	Enables the named event
EventCount	Returns the number of events configured
EventExists	Check if an event exists
GetEvent	Returns a reference to an event
GetEventEx	Get an event by name
GetEventList	Get all events
GetLastEvent	Returns the name of the last event that was triggered
NewEvent	Creates a new event with the given name
NewCondition	Creates a new condition that can be added to an event
NewDevFunc	Creates a new device function that can be added to an event
NewTimeEvent	Create a new event based on time
NewRecurringEvent	Create a new event based on a recurring time
RemoveDelayedEvent	Removes a queued device action or event
SaveEventsDevices	Tell HomeSeer to update events and devices
TriggerEvent	Force the actions of an event to execute

**EMAIL Related Functions**

<i>Function</i>	<i>Operation</i>
MailMsgCount	Returns the number of unread messages
MailFrom	Returns the from field of a message
MailFromDisplay	Get the actual name of the mail sender
MailDate	Returns the data the message was sent
MailSubject	Returns the subject of the message



MailText	Returns the body of the message
MailTo	Get the email address of the sender
MailToDisplay	Get the actual name of the sender
MailDelete	Delete a mail message
MailTrigger	Returns the index of the message that caused the last event trigger
SendEmail	Sends an email message

### CPU-XA/Ocelot Functions

These functions are referenced through the object cpuxa. IE: cpuxa.PlayIR 1

***Note that all these functions work with the CPU-XA, the Ocelot, and the Leopard touchscreen.***

<i>Function</i>	<i>Operation</i>
cpuxa.GetPoint	Get an input point
cpuxa.LearnIRStart	Start learning an IR command
cpuxa.LearnIREnd	Wait for IR learn to complete
cpuxa.PlayIR	Send a learned IR command
cpuxa.SendRaw	Send raw data to unit
cpuxa.SetVar	Set an internal variable
cpuxa.SetPoint	Set an output relay
cpuxa.PlayRemoteIR	Send an IR from a remote module

## 15.8 Function descriptions

### 15.8.1 Misc Functions

#### 15.8.1.1 CreateVar

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Name of variable
<u>Return value</u>	<u>Type</u>	<u>Description</u>
error code	string	Empty string if no error, else error string if variable already exists

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 a variant and can be used to hold any variable type including references to objects that are created with CreateObject.

#### Example

```
dim errst
errst = CreateVar("myvar")
if errst <> "" then
    msgbox "Error creating variable"
end if
```

#### See also

SaveVar  
GetVar  
DeleteVar

#### 15.8.1.2 DeleteVar

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Name of variable
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none	n/a	n/a

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

#### See Also

CreateVar  
GetVar  
SaveVar

### 15.8.1.3 SaveVar

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	name of variable
obj	object	object to be saved

<u>Return value</u>	<u>Type</u>	<u>Description</u>
error code	string	returns an empty string if no error occurred, else an error string is returned

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.

#### Example

```
dim errst
dim myvalue

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

#### See also

CreateVar  
GetVar  
DeleteVar

### 15.8.1.4 GetVar

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	name of variable

<u>Return value</u>	<u>Type</u>	<u>Description</u>
variable item	variant	returns the variable saved

Finds the variable associated with the name parameter and returns it.

#### 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  
SaveVar  
DeleteVar

### 15.8.1.5 ClearLog

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none	n/a	n/a

Clears the event log in memory. The event log file "ah.log" is untouched.

#### Example

```
hs.ClearLog
```

#### See Also

WriteLog

### 15.8.1.6 SetVolume

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
left	long	left channel volume level 0-20
right	long	right channel volume level 0-20
device (optional)	long	ID of audio output device. If this parameter is missing, device ID 0 is used.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none	n/a	n/a

Sets the volume of the WAVE device. This can be used to set the volume of the text-to-speech output. The volume level must be in the range between 0 and 20, where 20 is maximum volume and 0 is minimum volume.

#### Example

```
Sub Main()

    hs.SetVolume 20,20
    hs.speak "I am speaking louder",TRUE
    hs.SetVoume 5,5
    hs.speak "I am speaking softer",TRUE

end sub
```

### 15.8.1.7 GetVolume

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
channel	long	0=left channel and 1=right channel
device id	long	ID of device to get the volume from. 0 is normally the system sound card.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
volume level	long	The current volume level of the requested channel. Range is 0 -> 20

**15.8.1.8 version**

<u>Return value</u>	<u>Type</u>	<u>Description</u>
version string	string	the current version of HomeSeer

**Example**

```
sub main()

    dim s

    s=hs.version
    msgbox "The version of HomeSeer is "&s

end sub
```

**15.8.1.9 WaitSecs**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
seconds	integer	number of seconds to wait

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none	n/a	

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.

**Example**

```
sub main()

    hs.speak "hold on while I wait for 3 seconds"
    hs.WaitSecs 3
    hs.speak "ok, I'm done"

end sub
```

**See Also****See Also**

WaitEvents

**15.8.1.10 GetURL**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
host	string	name or IP address of host to connect to like "www.homeseer.com"
page	string	page to retrieve from server like "/news.htm"
strip_tags	Boolean	TRUE=strip HTML tags from returned page, FALSE=do not alter page
port	integer	port number on server to connect with (80=standard web server)

<u>Return value</u>	<u>Type</u>	<u>Description</u>
page contents	string	the contents of the requested web page. If an error occurs, the text "ERROR:" will be returned followed by a reason for the error.

Returns a web page. This is useful for retrieving pages like news and weather, and then having the agent speak the contents for you. This method can also be used to retrieve images such as jpeg or gif images.

To detect errors, check the return value, or call `hs.GetURLLastError` for an error code.

### Example

```
sub main()

dim page

page = hs.GetURL("www.homeseer.com", "/", TRUE, 80)
msgbox page

end sub
```

### 15.8.1.11 GetURLIE

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
host	string	name or IP address of host to connect to like "http://www.homeseer.com"
strip_tags	Boolean	TRUE=strip HTML tags from returned page, FALSE=do not alter page

<u>Return value</u>	<u>Type</u>	<u>Description</u>
page contents	string	the contents of the requested web page. If an error occurs, the returned string will be empty.

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.

### See Also

`GetURL`

### 15.8.1.12 ftp

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
host	string	name or IP address of host to connect to like "www.homeseer.com"
username	string	username for access to the server
password	string	password for access to the server
command	string	can be one of the following FTP commands: put, get, del, dir

path	string	path to the file like "public"
local_file	string	filename of file where the downloaded file will be saved
remote_file	string	filename of remote file on server to download

<u>Return value</u>	<u>Type</u>	<u>Description</u>
depends on command	string	for the "dir" command, a directory listing is returned. For all commands, if no error occurs and empty string is returned, else and error message is returned which starts with the text "ERROR".

Gives access to ftp servers. This command is used mostly for downloading files using the ftp protocol. For error checking see the function `hs.FTPLastError`.

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.

### Example

```
sub main()

    dim s
    dim host
    dim user
    dim password
    dim command
    dim rfile
    dim lfile
    dim path

    host = "homeseer.com"
    user = "anonymous"
    password = "user@company.com"
    command = "get"
    rfile = "remote_test.htm"
    lfile = "c:\remote_test.htm"
    path = "pub"
    ' get the file
    s = hs.ftp(host,user,password,command,path,lfile,rfile)

end sub
```

### See Also

`ftpSetProxy`

**15.8.1.13 FTPLastError**

<u>Return value</u>	<u>Type</u>	<u>Description</u>
error code	string	returns the last error string. If no error occurred, it returns an empty string.

Returns the last error from the last hs.ftp command.

**See Also**

ftp

**15.8.1.14 ftpSetProxy**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
phost	string	name or IP address of host to connect to like "www.homeseer.com"
puser	string	The username to log in with
ppassword	string	The password used to access the proxy
pport	integer	port number on server to connect with (80=standard web server)
ptype	integer	The proxy type, see types below

<u>Return value</u>	<u>Type</u>	<u>Description</u>
N/A		

Gives access to ftp servers. This command is used to set the proxy server and the proxy type in order to access a given server.

Proxy type values:

Value	Description
0 (none)	No proxy server is being used. This is the default value.
1 (user)	The client is not logged into the proxy server. The USER command is sent in the format username@ftpsite followed by the password. This is the format used with the Gauntlet proxy server.
2 (login)	The client is logged into the proxy server. The USER command is then sent in the format username@ftpsite followed by the password. This is the format used by the InterLock proxy server.
3 (open)	The client is not logged into the proxy server. The OPEN command is sent specifying the host name, followed by the username and password.
4 (site)	The client is logged into the server. The SITE command is sent, specifying the host name, followed by the username and the password.

**See Also**

ftp



### 15.8.1.15 SetRemoteTimeout

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
timeout	integer	Sets the number of seconds to wait for a remote host to respond when using hs.GetURL or hs.ftp If this function is never called, the remote timeout is set to 60 seconds

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

#### Example

```
' set the remote timeout to 30 seconds
hs.SetRemoteTimeout 30
```

### 15.8.1.16 GetAppPath

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
None		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
Application path	string	Returns the path to the HomeSeer install directory. Useful for finding HomeSeer specific files like the event log and script created files.

#### Example

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

### 15.8.1.17 GetIPAddress

Parameter	Type	Description
None		

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

#### Example

```
sub main()

    dim ipaddress

    ipaddress = hs.GetIPAddress
    hs.WriteLog "Info", "The IP Address is " & ipaddress

end sub
```

### 15.8.1.18 Keys

Parameter	Type	Description
key	string	Is the keycode to send (see below for special codes)
title	variant	Is the title string that appears in the main window of the target application you wish to control
wait	Boolean (optional)	Is true to slow down the sending of the keys, normally you want this to be TRUE, or (1).

This function allows you send keyboard commands to a running application. This is merely an interface into the Visual Basic *SendKeys()* 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 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 string.

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses ( ) have special meanings to *SendKeys*. 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:

Key	Code
-----	------

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

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:

Key	Code
SHIFT	+
CTRL	^
ALT	%

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.

### Example

This script will launch the calculator program:

```
sub main()

    dim I
    i=hs.launch("calc.exe", "")

end sub
```

This script will use the calculator to add some numbers:

```
sub main()

    hs.speak "I will add some numbers"
    hs.keys "1{+}2{+}3{ENTER}", "calc", 1

end sub
```

## 15.8.1.19 Launch

Parameter	Type	Description
name	string	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 pathname to the file. Application files can also be launched and the application that owns the file will be executed. For example, you could set the name to "c:\docs
parameters	string	Any parameters or command line switches that are to be passed to the application
direc	string	The working directory the application is launched from. Leave an empty string for most applications.

Return value	Type	Description
process instance	long	The instance number of the process, which is not very useful.

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

### 15.8.1.20 **Run**

Parameter	Type	Description
scr	string	Is the filename of the script to run. Do not include the path in the script name.

Runs another script.

Scripts must be located in the *scripts* directory in the HomeSeer application directory.

#### **Example**

```
sub main()
    hs.Run "lights_off.txt"
end sub
```

#### **See Also**

RunEx

### 15.8.1.21 **RunEx**

Parameter	Type	Description
scr	string	is the filename of the script to run. Do not include the path in the script name.
func	string	The name of the function to execute.
param	variant	A parameter to send to the function. This can be a string or numeric value.

Return value	Type	Description
	variant	Returns any value that the called script returns.

Runs a function in a script with a parameter. This will also return a value from the called script.

Scripts must be located in the *scripts* directory in the HomeSeer application directory.

#### **Example**

Consider the following script named *test.txt*:

```
launch_app(appname)
    hs.Launch appname
    ' now return some text to the caller
    launch_app = "The application is launched"
end sub
```

The above script could be called using the *RunEx* call from another script like:

```
sub main()  
  
    dim s  
    ' call the script and get the return value  
    s = hs.RunEx("test.txt","launch_app","notepad.exe")  
  
    ' display the returned string  
    msgbox s  
  
end sub
```

**See Also**

Run

**15.8.1.22 StringItem**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
str	string	Is the string to search in.
index	long	is the number of the string you wish returned. 1=string 1,2=string 2 etc.
sep	string	is the string that separates all other strings.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
string part	string	The part of the string requested

Retrieves the substring from a string. Given a string like "data;data1;data2", this function will retrieve the string at the specified index. Passing the index as 2, would return the string "data1".

**15.8.1.23 Sunrise**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
sunrise time	string	A string representing the time of sunrise.

Returns the time of Sunrise. Read only property.

**Example**

```
sub main()

    dim t

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

end sub
```

**See Also**

Sunset

**15.8.1.24 Sunset**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
sunset time	string	A string representing the time of sunset.

Returns the time of Sunset. Read only property.

**See Also**

Sunrise

**15.8.1.25 SetSecurityMode**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
mode	integer	0=disable and 1=enable.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Enables or disables security mode. See the HomeSeer help file for more information on the Security mode. Security mode varies the times of timed events.

**Example**

The following script statement will enable security mode.

```
hs.SetSecurityMode 1
```

**15.8.1.26 WriteLog**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
class	string	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".
message	string	Is the text to be displayed in the log, like a descriptive error message.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Write a message to the event log

**Example**

```
sub main()
    hs.WriteLog "Error","An error has occurred in my script!"
end sub
```

**15.8.1.27 WaitEvents**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

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.

### Example

```
sub main()

    dim x
    do

        ' allow other things to run
        hs.WaitEvents
        ' wait for an X10 command to be received
        x = hs.LastX10
        if x <> "" then exit do

    loop
    ' handle the x10 command here
    msgbox "The X10 command was: "&x

end sub
```

### See Also

WaitSecs

## 15.8.1.28 X10InterfaceStatus

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	integer	Returns one of the following integer values:  0 = No Error  1 = Error sending to device, usually a timeout or no response from the interface  2 = Init Error, the interface was not able to be initialized

Returns the status of the X10 interface. This can be used to determine if there is a connection or hardware problem with the configured X10 interface.

The status of other interface types may be obtained by calling the InterfaceStatus method on the interface object. See the plug-in collection object for more information.

### Example

```
sub main()

    dim status
```

```

status = hs.X10InterfaceStatus
hs.WriteLog "Info", "The X10 Interface status is: " & cstr(status)

end sub

```

### 15.8.1.29 WEBStatsPageViews

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
=value	integer	Set to 0 when clearing the stats.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
page statistics	integer	The number of page views from the HomeSeer web site as an integer.

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.

#### Example

```

' 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

```

### 15.8.1.30 WEBLoggedInUser

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
current user	string	Returns the last user who logged into the web server. Useful for scripts that may not want to run if a guest is logged in.

**15.8.1.31 WEBValidateUser**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
username	string	name of user to validate
password	string	password of user to validate
<u>Return value</u>	<u>Type</u>	<u>Description</u>
user authorization	Boolean	Returns TRUE if the given username / password pair is valid for the web server. Useful if you create your own login ASP web page.

**15.8.1.32 Ping**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
host name	string	Name or IP address of host to ping
<u>Return value</u>	<u>Type</u>	<u>Description</u>
host status	integer	Returns 0 if host is alive, else 26118 is returned indicating host is not available.

**15.8.1.33 System**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
system object	object	Returns a reference to the system object.

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. See the system functions section for more information.

**Example**

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

**15.8.1.34 SystemUptime**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	A string that is the system uptime.

Returns the amount of time HomeSeer has been running in the format:

```
days hours:minutes:seconds
```

**Example**

```
' Set a virtual device to display the system uptime
```

```

sub main()

    dim s
    s = hs.SystemUptime
    hs.SetDeviceString "z1", "Uptime: "&s

end sub

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

```

### 15.8.1.35 Shutdown

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
None		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Causes HomeSeer to shutdown immediately. Same affect as selecting file->exit from the file menu within HomeSeer. If HomeSeer Phone is running, that will be shutdown also.

### 15.8.1.36 PlayWavFile

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
filename	string	Complete path to the wave file to play

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

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

#### See Also

PlayWavFileEx

### 15.8.1.37 PlayWavFileEx

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
filename	string	Complete path to the wave file to play
deviceid	integer	ID of the wave device that will play the wave file. Useful if you want to play the wave file out an alternate sound card
volume	integer	Volume level to use when playing. Range is 0 to 100. Set the value to -1 if you want to use the currently set volume level
wait	Boolean	TRUE = do not return until the wav file has finished playing FALSE = play the wav file in the background. The function returns immediately

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Play a WAV file out a specific audio device. Also allows playing the WAV file in the background and setting the volume level.

#### See Also

PlayWavFile

### 15.8.1.38 ScheduleFile

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
filename	string	Complete path to a new configuration file to use

<u>Return value</u>	<u>Type</u>	<u>Description</u>
filename	string	Returns the full path and name of the current configuration file

This is a settable and readable property. Setting this property configures HomeSeer to use a new configuration file. Reading this property reports the currently configured config file. Configuration files hold all configured devices and events.

#### Example

```
' Set a new configuration file
hs.ScheduleFile = "c:\newconfig.xml"

' Read the current configuration file

dim config_file
config_file = hs.ScheduleFile
```

### 15.8.1.39 GetINISection

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
section	string	Name of section in INI file to get, like "Settings" for the HomeSeer settings section.
filename	string	filename of INI file to access. The HomeSeer INI file is "settings.ini". The filename is relative to the HomeSeer directory.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
ini section	string	Returns all values in the given INI section. Each section is separated with a NULL character.

#### Example

```
sub main()

dim items
```

```

dim section

section = hs.GetINISection("Settings", "settings.ini")

items = split(section,chr(0))
for I = 0 to ubound(items)
    hs.WriteLog "Item",item(i)
next

end sub

```

#### 15.8.1.40 GetINISetting

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
section	string	Name of section in INI file to get, like "Settings" for the HomeSeer settings section.
key	string	name of key in INI file to access
default	string	default value to return if the key is not found
filename	string (optional)	filename of INI file to access. The HomeSeer INI file is "settings.ini". The filename is relative to the HomeSeer directory. If this parameter is omitted, the HomeSeer settings.ini file is used.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
ini key value	string	Returns the value associated with the requested key from an INI file.

#### Example

```

sub main()

dim value

' get the current config file in use by HomeSeer
value = hs.GetINISetting("Settings","configfile","")

msgbox "The HomeSeer config file is: " & value

end sub

```

**15.8.1.41 ClearINISection**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
section	string	Name of section in INI file to clear, like "Settings" for the HomeSeer settings section.
filename	string	filename of INI file to access. The HomeSeer INI file is "settings.ini". The filename is relative to the HomeSeer directory.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Clears an entire section in an INI file. If the "Settings" section is cleared in the settings.ini file, HomeSeer will be set to its default settings.

**15.8.1.42 SaveINISetting**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
section	string	Name of section in INI file to save to, like "Settings" for the HomeSeer settings section.
key	string	name of key in INI file to access
value	variant	value to save in the given key
filename	string (optional)	filename of INI file to access. The HomeSeer INI file is "settings.ini". The filename is relative to the HomeSeer directory. If this parameter is omitted, the HomeSeer settings.ini file is used.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
ini key value	string	Returns the value associated with the requested key from an INI file.

Saves a key/value pair in an INI file.

**Example**

```
hs.SaveINISetting "my settings", "zip code", "03110", "my_settings.ini"
```

**15.8.1.43 GetPlugins**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
plug-in collection	collection	Returns a collection of all objects, one object for each plug-in installed

**Example**

```
sub main()
```

```

dim plug-in

dim item

set plug-in = hs.GetPlugins

for each item in plug-in

    hs.WriteLog "Item","Plugin name is: " & item.name

next

end sub

```

#### 15.8.1.44 plugin

Parameter	Type	Description
plug-in specific		

Return value	Type	Description
plug-in specific		

Direct access to a plug-in. Some plug-ins may offer special properties and methods that are not available through the standard plug-in interface. This object allows direct access to a specific plug-in. The plug-in object is a collection of all installed plug-in's indexed by the plug-in name. One common interface function is the plug-in name. The following example gets the name from the Applied Digital Ocelot plug-in:

```

dim b_name

b_name = hs.plugin("Applied Digital Ocelot").name

```

#### 15.8.1.45 GetPrinter

Parameter	Type	Description
none		

Return value	Type	Description
printer object	object	Returns a reference to the printer object. This is the VB printer object which can be used to set various properties for printing to the system printer

#### Example

```

sub main()

' set the font size for printing

dim printer
set printer = hs.GetPrinter
printer.FontSize = 12
hs.PrintOut "I am printing this to the printer"

end sub

```



**15.8.1.46 PrintOut**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
text	string	Text to print on the system printer

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Sends the given text to the system printer. See the function GetPrinter for an example.

**15.8.1.47 IsScriptRunning**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
script name	string	Name of script to check.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
true or false	Boolean	Returns TRUE if the specified script is currently running, else returns FALSE.

**15.8.1.48 NoLog**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
log status	boolean	Enables disables the logging of X10 sent messages in the log.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
true or false	boolean	Returns TRUE if logging is disabled, else FALSE if logging is disabled.

If a scripts uses hs.ExecX10 to send X10 signals, the log may become cluttered with X10 messages. Use this property to disable logging. This setting is only in affect as long as the current script is running. When the script exits, logging is again re-enabled.

**Example**

```
' disable logging
hs.NoLog = TRUE
' enable logging
hs.LoLog = FALSE
```

**15.8.1.49 ControlThermostat**

<b>Parameter</b>	<b>Type</b>	<b>Description</b>
device	string	Device code of thermostat like "J1". This is configured device code.

<b>Return value</b>	<b>Type</b>	<b>Description</b>
command	string	The the thermostat command like "GetTemp". See below for possible commands.
value	value	Value to set if setting temperature, mode, or fan.

This function gives direct access to thermostats. Thermostats are supported with special .THM scripts. The command parameter is the entry function that is called. The functions available are:

GetTemp = Returns the current temperature  
 GetSetPoint = Returns the current heat set point  
 GetFan = Returns the Fan mode (0=off 1=on)  
 GetMode = Returns the operating mode (0=off 1=heat 2=cool 3=auto)  
 SetSetPoint = Sets the current heat set point  
 SetFan = Sets the current fan mode  
 SetMode = Sets the current operation mode  
 SetCoolSetPoint = Sets the current cool set point (if supported by thermostat)

**Example**

```
' get the current temperature

dim temp

temp = hs.ControlThermostat("J1","GetTemp",0)

' set the set point to 72 degrees

hs.ControlThermostat "J1","SetSetPoint",72
```

## 15.8.2 Media Functions

The following functions can be used to access the windows media player. With these functions you can instruct the media player to play any type of audio file it supports. The media player functions only access Media Player 6.4. For access to Media Player 7 and later, check out our Media Player plug-in.

### 15.8.2.1 MediaMute

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
mute	boolean	TRUE = mute the selection, FALSE = unmute the selection.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Mutes the currently playing media selection. The selection continues to play, but not sound is heard.

#### Example

```
sub main()
    ' mute the media player
    hs.MediaMute FALSE
end sub
```

### 15.8.2.2 MediaPlay

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Starts playing the selection as specified with the hs.MEDIAFilename property.

### 15.8.2.3 MediaVolume

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
level	integer (property)	Sets the volume level. 0=full volume and -3000 is the lowest volume.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Read/Write property. Sets and gets the current volume level of the playing media selection.

#### Example

```
sub main()
```

```

' get the current volume level
dim level
level = hs.MediaVolume

' set the volume to full
hs.MediaVolume = 0

end sub

```

### 15.8.2.4 MediaPause

Parameter	Type	Description
none		

Return value	Type	Description
none		

Instructs the Windows Media Player to pause the currently playing selection. The selection may be resumed by calling the `hs.MediaPlay` function.

### 15.8.2.5 MEDIAFilename

Parameter	Type	Description
filename	integer	sets the filename of the media selection to play. The filename may be any valid file supported by the Windows Media Player.

Return value	Type	Description
none		

Read/Write property. Sets the filename or URL that is to played using the Windows media player. Call `MEDIAPlay` to actually start playing the selection.

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

Scripts must be located in the *scripts* directory in the HomeSeer application directory.

### 15.8.2.6 MedialsPlaying

Parameter	Type	Description
none		

Return value	Type	Description
		TRUE = a media selection is currently playing and the sound card is most likely busy FALSE = a media selection is not playing, and the sound is most likely free

Checks if the media player is currently playing a selection. Returns TRUE if so else FALSE if not.

### 15.8.2.7 **MediaStop**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Instructs the Windows Media Player to stop playing the current selection.

## 15.8.3 Communications Port Functions (RS232)

### 15.8.3.1 OpenComPort

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
port	integer	The port number to open. An error is returned if the port is already open or is not installed on the system.
config	string	See below.
mode	integer	See below.
cb_script	string	See below.
cb_func	string	See below.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
		The function returns an empty string if it was successful, otherwise it returns a text string describing the error.

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. The port is closed when the application terminates however.

#### **port**

The port number to open. An error is returned if the port is already open or is not installed on the system.

#### **config**

Is composed of four settings and has the following format:

```
"BBBB , P , D , S"
```

Where 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 following table lists the valid baud rates.

```
110
300
600
1200
2400
4800
9600
14400
19200
28000
38400
56000
57600
115200
```

128000  
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

#### **mode**

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

0 = raw mode

In this mode, each character that is received on the com port causes the specified script and specified 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 CRLF pair 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 a CRLF.

#### **cb\_script**

This 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
```

#### **cb\_func**

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.

#### **See Also**

`GetComPortData`  
`SendToComPort`  
`GetComPortCount`  
`CloseComPort`

### 15.8.3.2 CloseComPort

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
port	integer	The number of the port to be closed.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Closes a communications port previously opened with OpenComPort.

#### See Also

OpenComPort

### 15.8.3.3 GetComPortCount

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
port	integer	The port number of the port to check.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	integer	The number of characters available at the COM port.

Returns the number 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.

#### See Also

OpenComPort

### 15.8.3.4 GetComPortData

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
port	integer	The COM port to read.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	variant	The data available is a string of characters.

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  
GetComPortCount



### 15.8.3.5 **SendToComPort**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
port	integer	The COM port to send the data on. An error is returned if the port is not already opened.
data	string	The actual data to send out the COM port. This is a string.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	

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

**See Also**

OpenComPort

### 15.8.3.6 **SetComPortRTSDTR**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
port	integer	The com port to access. Valid range is 1 to 8.
rts_val	boolean	Set to TRUE to raise the RTS line, else set to FALSE to lower the line
dtr_val		Set to TRUE to raise the DTR line, else set to FALSE to lower the line
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

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

**See Also**

OpenComPort

## 15.8.4 Infrared Functions

### 15.8.4.1 SendIR

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
ir	string	Is a string of I/R text commands. These are the same commands that are listed for each of the keys for a particular device.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Send an I/R command to an attached infrared device.

This example sets the "vcr" to "record"

#### example

```
hs.SendIR "VCR,Record"
```

## 15.8.5 Voice Related Functions

### 15.8.5.1 ClearAllVoiceCommands

Parameter	Type	Description
none		

Return value	Type	Description
none		

Clears all voice commands that were added with AddVoiceCommand.

#### See Also

AddVoiceCommand  
System Functions

### 15.8.5.2 LastVoiceCommand

Parameter	Type	Description
none		

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

Get the last voice command recognized. Read only property.

#### Example

If a voice command was set to:

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

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

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

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

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

```
sub main()
    dim v
    v=hs.LastVoiceCommand
```

```
msgbox "I heard "&v
```

```
end sub
```

### See Also

System Functions

#### 15.8.5.3 LastCommandSelected

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
event name	string	Returns the name of the event that was triggered by the last voice command

Get 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. Read only property.

#### 15.8.5.4 AddVoiceCommand

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
cmd	string	Voice command to add

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	

This will add the given 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
```

### Example

The following script will read your email messages. This script is available in the scripts directory as "read\_messages.txt"

```
sub main()

dim s
dim c
c=hs.MailMsgCount
hs.speak "you have "&cstr(c)&" messages"
' if no messages then just exit
if c=0 then

hs.speak "I guess no one wants to talk to you right now. Maybe later."
```

```
        exit sub

    end if

    hs.speak "would you like me to read them to you?"

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

    ' create our own private recognition list
    hs.AddVoiceCommand "yes"
    hs.AddVoiceCommand "no"

    ' wait for a voice command
    do

        s=hs.LastVoiceCommand
        if s<>" " then exit do
        hs.WaitEvents

    loop

    ' check the actual command
    if s<>"yes" then

        hs.speak "ok"
        exit sub

    end if

    ' read the messages
    for i=0 to c-1

        hs.speak "message "&cstr(i+1)
        hs.speak hs.MailDate (i)
        hs.speak "message is from,, "&hs.MailFrom(i)
        hs.speak "the subject of the message is,,"&hs.MailSubject(i)
        'hs.speak hs.MailText(i)

    next

    ' HomeSeer Phone really needs to be told that we are done
    ' using the voice commands
    system.ClearAllVoiceCommands

end sub
```

**See Also**

ClearAllVoiceCommands  
System Functions

### 15.8.5.5 ListenForCommands

Parameter	Type	Description
action	boolean	TRUE = listen for event name commands, FALSE = listen only for the attention phrase
Return value	Type	Description
none		

This will switch the computer from either listening for event name commands or listening for the attention phrase. This function only works if HomeSeer is in *Always Listen* mode, rather than using MSAgent to listen.

#### Example

```
sub main()

' listen only for attention phrase
hs.ListenForCommands FALSE

end sub
```

### 15.8.5.6 SetSpeaker

Parameter	Type	Description
name	string	
Return value	Type	Description
	string	

Sets the speaker for voice recognition. You can train the system for different speakers. Use this function to set the name of the speaker, then run the training wizard to train the system for this person. Recognition will function better if it knows who the speaker is.

### 15.8.5.7 StartListen

Parameter	Type	Description
none		
Return value	Type	Description
none		

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.

#### See also

StopListen  
System Functions

### 15.8.5.8 StopListen

Parameter	Type	Description
none		

Return value	Type	Description
none		

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.

#### See also

StartListen  
System Functions

### 15.8.5.9 Speak

Parameter	Type	Description
text	string	Is the string you want to speak. It may also be the complete path to a WAV file, and the WAV file will be played.

wait	boolean (optional)	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 system. If this parameter is missing, the system will not wait.
------	--------------------	---

Return value	Type	Description
none		

Speak some text.

#### Example

```
sub main()
    ' speak and wait
    hs.speak "hello there", TRUE
end sub
```

#### See Also

System Functions  
SpeakToFile

**15.8.5.10 StopSpeaking**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Causes any speaking to stop immediately.

**Example**

```
hs.StopSpeaking
```

**15.8.5.11 SpeakToFile**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
text	string	Is the string you want to speak.
voice	string	The name of the voice you want to use for speaking. This string must match the voice name exactly. Voice names can be found on the text-to-speech tab in the HomeSeer options. If the name is omitted, the default voice as specified in the computers speech control panel is used. Only SAPI 5 voices may be used with this function. SAPI 5 voices contain a "(V5)" in their name. The "(V5)" text may be omitted when the name is passed to this function.
filename	string	Full path to the file where the voice output will be saved.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Speak some text and save the result in a WAV file.

**Example**

```
sub main()
    hs.SpeakToFile "Hello from a file!", _
    "ATT DTNV 1.3 Crystal", "c:\voice.wav"
end sub
```

**See Also**

Speak



## SpeakEx

### 15.8.5.12 **SpeakEx**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	integer	The device number of the output device. Device 0 is usually the computer speakers and the default sound card.
text	string	Is the string you want to speak.
wait	boolean	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 system. If this parameter is missing, the system will not wait.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

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

#### See Also

Speak  
SpeakToFile

### 15.8.5.13 **MuteSpeech**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
=TRUE	boolean	speech output is silenced.
=FALSE	boolean	speech output is enabled.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Read/Write property. 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 sourced from scripts.

The status bar in the main HomeSeer display window will display *Mute* while speech is muted.

#### 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

**15.8.5.14 ListenMode**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
current listening state	integer	Returns the current listen mode which is define as: 1 = Not Listening 2 = Listening for commands 3 = Listening for attention

## 15.8.6 Device Functions

### 15.8.6.1 DeviceStringByName

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Is the name of the device. The name includes its location like: <i>den table lamp</i>
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Returns the character string set for a device. See SetDeviceString().

**See also**

DeviceString

### 15.8.6.2 SetDeviceStringByName

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Is the name of the device including its location like <i>den table lamp</i> . Note the name is not case sensitive.
message	string	Is the status string for the device, like <i>72 degrees</i>
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Sets a string as the device status using the actual name of the device combined with its location. See SetDeviceString()

**See also**

SetDeviceString

### 15.8.6.3 NewDevice

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Is the name of the new device.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	integer	Index of device that may be used in subsequent calls to <i>GetDevice</i> .

Creates a new device and gives it the name given. The new device has the housecode and unitcode set to "A1", and all other attributes of the device are cleared. The device has no module type and has no location.

**See also**

GetDevice

### 15.8.6.4 NewDeviceEx

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Is the name of the new device.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
reference to DeviceClass	object	Returns a reference to the actual DeviceClass object. This allows direct access to all the properties of a device

Creates a new device and sets its name property to the name given. The housecode and unitcode of the device are set to A1. The returned object is a reference to the new device object. See the description for the GetDevice function for a list of the properties available.

#### Example

```
sub main()
    dim dv
    set dv = hs.NewDeviceEx("my device")
    dv.location = "living room"
end sub
```

#### See also

NewDevice  
GetDevice

### 15.8.6.5 GetDevice

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	

<u>Return value</u>	<u>Type</u>	<u>Description</u>
DeviceClass	object	A reference to a device object of type <i>DeviceClass</i> .

This function will return a reference to a specific device. The function returns a reference to a DeviceClass object. The object is what HomeSeer uses internally to represent a device. With this object you can set and get all properties of a device.

The index parameter is the index into the device list of the device you wish to retrieve. You can use the **DeviceCount** function to retrieve the total number of devices that have been configured. The GetDeviceEx function can also be used to get a reference to a device using it's name.

The device class has the following properties:

<u>Property</u>	<u>Type</u>	<u>Description</u>
<b>can_dim</b>	boolean	true if device dimmable
<b>dc</b>	string	device code as a string (ie: "1")
<b>hc</b>	string	house code as string (ie: "A")
<b>location</b>	string	a string describing the location of the device

<b>dev_type_string</b>	string	a string that is the name of the last device type chosen for this device
<b>name</b>	string	a string, which is the name of the device
<b>ref</b>	long	a device reference used internally. This is unique for every device. Events hold this as a reference to the device. If the device is deleted, all events that reference the device will no longer control it.
<b>status_support</b>	boolean	true if the device supports the X10 status request command
<b>misc</b>	long	a bit field that holds various properties for the device. The field is a long value. The bits are defined as: &h1 = device supports the X10 preset dim command &h2 = device supports the X10 Extended dim command &h4 = smart linc switch &h8 = no logging &h10 = status only device &h20 = hide device from views &h40 = device is a thermostat &h80 = include device in power fail &h100 = value names are selectable from web and Win UI (values assigned with hs.DeviceValuesAdd)
<b>values</b>	string	Holds the name/value pairs as added with the hs.DeviceValueAdd function. See the plug-in SDK for more information.
<b>buttons</b>	string	Holds information about buttons added with hs.DeviceButtonAdd. See the plug-in SDK for more information.

### Example

Here is a sample script that changes the location name for a device.

```
' this script will retrieve the device properties for the device
' named "test device"
' and set its location to "living room"
'
sub main()

dim dev_count
dim I
dim device
dev_count = hs.DeviceCount          ' get total number of devices
for i=1 to dev_count

    set device = hs.GetDevice(i)
    if device.name = "test device" then
        device.location = "living room"
    end if
next

end sub
```

### 15.8.6.6 GetDeviceEx

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Name of device. The name consists of the devices location and name, like "living room lamp". The search is not case sensitive.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
DeviceClass	object	A reference to a device object of type <i>DeviceClass</i> .

This function will return a reference to a specific device. The function returns a reference to a DeviceClass object. The object is what HomeSeer uses internally to represent a device. See the GetDevice function for the list of properties of the device object.

#### Example

```
sub main()

    dim dv

    set dv = hs.GetDeviceEx("living room lamp")
    if dv is nothing then

        msgbox "Error, device not found"

    else

        ' access device object here
        hc = dv.hc      ' housecode
        dc = dv.dc      ' unit code

    end if

end sub
```

### 15.8.6.7 DeviceCount

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
count of devices	integer	The number of devices configured.

Returns the total number of devices currently configured.

#### Example

```
dim count
count = hs.DeviceCount
```

#### See also

GetDevice

### 15.8.6.8 **GetDeviceCode**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Is the name of the device including its location like "den table lamp"
<u>Return value</u>	<u>Type</u>	<u>Description</u>
X10 housecode / unitcode	string	The X10 device code for the device like "A1". If the device does not exist, an empty string is returned.

Returns the X10 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 X10 device code.

#### **Example**

```
dim code
code = hs.GetDeviceCode("den table lamp")
msgbox "The housecode unitcode is: " & code
```

### 15.8.6.9 **PollDevice**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	Is the housecode unitcode of the device to be polled, like "A1".
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

This function will send an X10 Status Request command to the given device. The function will wait up to 8 seconds for the device to reply, then return. The status of the device may be read using the DeviceStatus function. The status of the device will be updated in the user interface.

#### **Example**

```
dim status

hs.PollDevice "A1"

status = hs.DeviceStatus("A1")

if status = 2 then

    msgbox "The device is ON"

else if status = 3 then

    msgbox "The device is OFF"

end if
```



**15.8.6.10 ExecX10ByName**

<b>Parameter</b>	<b>Type</b>	<b>Description</b>
device	string	Is the name of the device along with the location separated by a space. If the device is name "light" and its location is "den", then device would be given as den light
cmd	string	Is the name of the X10 command as a string. The "No Cmd" command may used to send X10 address information only. No X10 command will be sent for this command. The command names are NOT case sensitive and are:  All Units Off All Lights On On Off Dim Bright All Lights Off Extended Hail Request Hail Ack Preset Dim Ex Data Xfer Status On Status Off Status Request Dim to Off No Cmd
dimval	integer (optional)	Is the % dim for "Dim" and "Bright" commands. For the "Pre-set Dim" command, this is a value between 0 and 31.
data2	integer (optional)	data byte for use with the Extended X10 command
wait	Boolean (optional)	if TRUE, the function will not return until the command is actually sent. If false or missing, the command will be queued for transmission.

<b>Return value</b>	<b>Type</b>	<b>Description</b>
status	integer	0 if device not found else 1.

Send an X10 command using the name of the device.

**Example**

Here is an example that dims the device named "light" that is located in the "den" to 40%:

```
hs.ExecX10ByName "den light", "Dim", 40
```

**See Also**

ExecX10

### 15.8.6.11 ExecX10

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
code	string	Is a string of X10 devices like "A1" or "A1+2+3+9"
cmd	string	Is the name of the X10 command as a string. The "No Cmd" command may used to send X10 address information only. No X10 command will be sent for this command. The command names are <i>NOT</i> case sensitive and are:  All Units Off All Lights On On Off Dim Bright All Lights Off Extended Hail Request Hail Ack Preset Dim Ex Data Xfer Status On Status Off Status Request Dim to Off No Cmd
dimval	integer (optional)	Is the dim amount as defined in the ExecX10ByName function.
data2	integer (optional)	Is the second data byte for the "Extended" X10 command. The <i>dimval</i> parameter is the first data byte if the command is "Extended"
wait	boolean (optional)	If TRUE, the command will not return until after the X10 command has actually been sent on the power line.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	integer	1 if successful else 0 if error

Execute an X10 command using the actual house code and device code.

#### Example

This example will turn on the device at code *B16*:

```
hs.ExecX10 "B16", "on", 0, 0
```

Some X10 devices allow you to program the device using just X10 addresses. HomeSeer can send just an address by setting the command to "no cmd". Here is an example that sends just the address *B2*:

```
hs.ExecX10 "B2", "no cmd", 0, 0
```

#### See Also

ExecX10ByName

### 15.8.6.12 **IsOnByName**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device name	string	Is the name of the device. The device name must include the device's location and it's name like "den table lamp". The name is not case sensitive.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	boolean	TRUE if a device is ON (or dimmed), else returns FALSE.

Checks the status of a device by name.

#### **See Also**

IsOn  
IsOff  
IsOffByName

DeviceStatus  
SetDeviceStatus

### 15.8.6.13 **IsOffByName**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device name	string	Is the name of the device. The device name must include the device's location and it's name like "den table lamp". The name is not case sensitive.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	boolean	TRUE if a device is ON (or dimmed), else returns FALSE.

Checks the status of a device by the device name.

#### **See Also**

IsOn  
IsOff  
IsOnByName

DeviceStatus  
SetDeviceStatus

#### 15.8.6.14 **IsOff**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	Is the X10 housecode unitcode of the device like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	boolean	TRUE if the device is OFF, else returns FALSE.

Checks the status of a device.

#### **See Also**

IsOn  
IsOnByName  
IsOffByName

DeviceStatus  
SetDeviceStatus

### 15.8.6.15 **IsOn**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	Is the X10 housecode unitcode of the device like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	boolean	TRUE if the device is ON, else returns FALSE.

Checks the status of a device.

#### **Example**

```
sub main()  
  
  if hs.IsOn("A1") then  
  
    hs.speak "the light is on"  
  
  end if  
  
end sub
```

#### **See Also**

IsOff  
IsOnByName  
IsOffByName

DeviceStatus  
SetDeviceStatus

**15.8.6.16 DeviceStatus**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	The actual device code like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
device status	integer	The return value is an integer of one of the following:  2 = ON 3 = OFF 4 = DIM 17 = UNKNOWN

Returns the actual status of the device.

**See also**

SetDeviceStatus  
IsOn  
IsOff  
IsOnByName  
IsOffByName

**15.8.6.17 SetDeviceStatus**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	Is the X10 housecode unitcode of the device like "A1"
status	integer	Is the status to set the device to. This integer value should be one of the following:  0 = All Units Off 2 = ON 3 = OFF 4 = DIM

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Sets the status of a device, whether the device is ON, OFF, or DIMMED.

**15.8.6.18 DeviceValue**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	The X10 housecode unitcode of the device like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	long	The value stored for the device, which is usually the dim level.

Returns the value stored for this device. Device values are a long value that is associated with



a device. This is a general-purpose value that you can set and read. Normally, the device value holds the dim level of an X10 device.

**See Also**

SetDeviceValue

DeviceValueByName

### 15.8.6.19 SetDeviceValue

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	Device code like "A1".
value	long	A numeric value like 50.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

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 an X10 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), as these devices will never be affected by X10 power line activity.

#### Example

```
sub main()  
    ' set the dim value of device B2 to 60%  
    hs.SetDeviceValue "B2",60  
end sub
```

#### See Also

DeviceValue

DeviceValueByName

**15.8.6.20 SetDeviceString**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	Is the X10 housecode unitcode of the device like "A1"
message	string	Is the status string for the device, like "72 degrees"
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Sets a string as the device status. The string "message" is displayed in the device status column. 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 X10 status 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.

Note that the HTML tags are stripped from the text when the status is displayed in the windows interface. The HTML will only appear on the HomeSeer web page.

**Example**

```
sub main()

  hs.SetDeviceString "A1",Motion Detected"

  ' add some HTML to the text to create a scrolling status

  hs.SetDeviceString "A2", "<MARQUEE><blink><b>Motion Detected</MARQUEE>
  </b></blink>"

end sub
```

**See Also**SetDeviceStringByName  
DeviceString**15.8.6.21 DeviceString**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
device string	string	the display string associated with the device

Returns the character string set for a device. See SetDeviceString().

**Example**

```
sub main()  
  
    dim s  
  
    s=hs.DeviceString("A1")  
    msgbox s  
  
end sub
```

**See Also**

SetDeviceString

### 15.8.6.22 DeviceValueByName

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device_name	string	The name of the device must contain both the location and name. If the device was name <i>lamp</i> , and its location was <i>living room</i> , the <i>device_name</i> parameter would be <i>living room lamp</i> .

<u>Return value</u>	<u>Type</u>	<u>Description</u>
device value	long	returns the value associated with the device

Same as DeviceValue, except you pass this function the *text name* of the device.

#### **See also**

DeviceValue

---

DeviceValueByName

### 15.8.6.23 SetDeviceValueByName

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device_name	string	The name of the device must contain both the location and name. If the device was name <i>lamp</i> , and its location was <i>living room</i> , the <i>device_name</i> parameter would be <i>living room lamp</i> .
value	long	
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Same function as SetDeviceValue except you pass this function the actual text name of the device.

**See also**

SetDeviceValue

**15.8.6.24 DeviceTime**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	The X10 housecode unitcode of the device like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
device time	long	Time since last device status change.

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. The time is in reference to when the computer started. For a more accurate value, use DeviceLastChange instead.

**See Also**

DeviceLastChange

**15.8.6.25 SetDeviceTime**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	The X10 housecode unitcode of the device like "A1"
value	long	Time in minutes since the device last changed status

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Sets the minute timer for a device, useful if you use a device as an elapsed time counter. Note that hs.DeviceLastChange may be more useful as it's accurate to the second as opposed to the minute.

**15.8.6.26 DeviceLastChange**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	The X10 housecode unitcode of the device like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
device time	date	Date and time the device last changed status. This saved in the device_state.txt file and restored when HomeSeer is started.

**Example**

```
sub main()

dim code
dim last_change

' get the last change time for the device name "living room lamp"

' get the X10 code for the device
code = hs.GetDeviceCode("living room lamp")
```

```
last_change = hs.DeviceLastChange(code)

end sub
```

**See Also**



DeviceTime

**15.8.6.27 SetDeviceLastChange**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	The X10 housecode unitcode of the device like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
change time	date	Date and time the device last changed status. This is saved in the device_state.txt file and restored when HomeSeer is started.

Sets the last change time of a device.

**Example**

```
hs.SetDeviceLastChange "A1",now
hs.SetDeviceLastChange "A1","4:00 PM"
```

**15.8.6.28 DeviceExists**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	The X10 housecode unitcode of the device like "A1"

<u>Return value</u>	<u>Type</u>	<u>Description</u>
status	integer	Returns -1 if the device does not exist, else it returns the index number of the device. The index number can then be used with GetDevice

**15.8.6.29      GetDeviceList**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
list of devices	collection	Returns a reference to HomeSeer's device list. HomeSeer stores devices in a collection. Each object of the collection is an object of type DeviceClass

**Example**

```

sub main()

dim devices

set devices = hs.GetDeviceList

' cycle through all devices and get their locations

for each dv in devices
    hs.WriteLog "Info", "Device location is: " & dv.location
next

end sub

```

**15.8.6.30      GetDeviceByRef**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
ref	long	The reference ID of a device

<u>Return value</u>	<u>Type</u>	<u>Description</u>
device	object as DeviceClass	Returns a reference to the given device object. If the device does not exist, then an empty reference is returned.

**15.8.6.31      LastX10**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
X10 command	string	returns the last X10 command received

Returns a string that contains the last X10 command received. If the script containing this function is run as the result of an X10 command received, this function will return the X10 command that caused the trigger.

The returned string has the following format:

*Housecode;device;command;extra1;extra2*

Housecode is the actual housecode the command was seen on like "A"

Device is the device the command was addressed to like: "3"

**Note:** Version 1.5 of HomeSeer returned multiple devices in the device parameter, like "1+2". To make parsing easier, version 1.6 now only returns a single device. If a script is called based on X10 received, then it will be called multiple times, once for each unit code.

Command is the X10 command number like: "2" for the X10 command ON

*Extra1* is the dim level for DIM and BRIGHT commands. For preset dim and extended dim commands, extra1 is the actual dim level for the command (0-31 for preset dim) (0-63 for extended dim)

*Extra2* is the second data byte received for the X10 extended command

The X10 command numbers are:

0	All Units Off
1	All Lights On
2	On
3	Off
4	Dim
5	Bright
6	All Lights Off
7	Extended
8	Hail Request
9	Hail Ack
10	Preset Dim
11	Ex Data Xfer
12	Status On
13	Status Off
14	Status Request
15	Dim to Off

### 15.8.6.32 ClearLastX10

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Clears the last X10 command received to an empty string. It is possible to go into a loop calling the LastX10 function to wait for an X10 command. If this is done, be sure and call WaitEvents() so the script engine does not think the script is locked up.

### 15.8.6.33 DeviceTimeByName

Parameter	Type	Description
device name	string	The name of the device must contain both the location and name. If the device was name <i>lamp</i> , and its location was <i>living room</i> , the <i>device name</i> parameter would be <i>living room lamp</i> .
Return value	Type	Description
	long	Time in minutes since last device status change.

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.

#### See Also

DeviceLastChange

### 15.8.6.34 RegisterStatusChangeCB

Parameter	Type	Description
script	string	Is the filename of the script to run. Do not include the path in the script name, the script is assumed to be in the scripts directory.
function	string	The function in the script to run, such as <i>main</i>
Return value	Type	Description
none		

HomeSeer has the ability to trigger events based on the status of a device changing. This is set in the event properties for the event. It may be useful to run a script when a device changes status. The *RegisterStatusChangeCB* can be used to register your script with HomeSeer. When a device changes status, your script will be called. The script is passed the *housecode* and *unitcode* of the device that changed status, as well as the status 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) = housecode of the device that changed status
parm(1) = devicecode of the device that changed status
parm(2) = the x10 status that the device changed to. See here for the codes.
```

parm(3) = the device reference number. Can be used with GetDevice 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.

### Example

```
' register a callback script

sub main()

    hs.RegisterStatusChangeCB "stat_change.txt", "statuscb"

end sub

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

sub statuscb(parm)

    dim hc
    dim dc
    dim status

    hc = parm(0)
    dc = parm(1)
    status = parm(2)

end sub
```

#### 15.8.6.35 UnRegisterStatusChangeCB

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Removes a script associated with a status change as set with RegisterStausChangeCB.

### 15.8.6.36 DeviceButtonAdd

Parameter	Type	Description
device	string	Device to add a button to like "A1"
ref	string	plug-in name or script call. For scripts, the format is: <script name>(<function name>,<parameter>),<button label>
name	string	The name to label the button
Return value	Type	Description
none		

This call allows for custom buttons to be added to devices. The buttons appear on the web status page, and in the menu when you right click a device in the Windows UI. The "ref" parameter can be a plug-in name, and would therefore be used by a plug-in itself. See the plug-in SDK documentation on using this call in a plug-in.

For scripts, a script function can be added as the action when a user presses the button. Each button can have it's own script. The script call includes a script name and function to call. For example, if the script to call has a function named "speak", and the parameter is the text to speak, and button should be labeled "speak now", then the "ref" parameter would be:

```
"speak.txt("speak","hello"),"speak now"
```

Each time this function is called, a new button is added to the device. You can add as many buttons as you like. HomeSeer saves the button configuration in the XML configuration file, so only call this function when you install the script.

#### Example

```
sub main()

' add a button to the device "A1" that allows a message to be
' spoken when the button is pressed. Label the button "speak now"

hs.DeviceButtonAdd "A1","speak.txt("speak","hello"),"speak now"

end sub

' the speak.txt script looks like

sub speak(speak_string)

hs.speak speak_string,false

end sub
```

#### See Also

DeviceButtonRemove

### 15.8.6.37 [DeviceButtonRemove](#)

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
device	string	Device to add a button to like "A1"
name	string	Button label of the button to remove

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Removes buttons added to a device using `hs.DeviceButtonAdd`

#### **See Also**



## DeviceButtonAdd

## 15.8.7 Event Functions

### 15.8.7.1 GetLastEvent

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
last event	string	A text string that is the name of the event.

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.

#### Example

```
sub main()
    dim t
    t = hs.GetLastEvent
    msgbox "This script is run from the event: " & t
end sub
```

### 15.8.7.2 TriggerEvent

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	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. Events "Evening" and "evening" would be considered the same. If there were duplicate event names, only the first one found would run.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	integer	1 if error, else 0 if no error. If an error is detected, and error message is written to the event log.

Force the trigger of an event.

#### Example

```
'Trigger the event named "turn all lights on"
sub main()
    hs.TriggerEvent "turn all lights on"
end sub
```

### 15.8.7.3 DeleteEvent

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
evname	string	Is the event name to delete. Note that the name is not case sensitive.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Deletes the event with the given name.

### 15.8.7.4 DisableEvent

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
evname	string	Is the event name to disable. Note that the name is not case sensitive.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Marks an event as disabled. All triggers are suspended until the event is re-enabled.

**See also**

EnableEvent

### 15.8.7.5 EnableEvent

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
evname	string	Is the event name to enable. Note that the name is not case sensitive, and the event must have already been disabled.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Marks an event as enabled. All triggers are active.

**See also**

DisableEvent

### 15.8.7.6 NewTimeEvent

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
evname	string	Is the name that will be given to the new event.
when	date	Is the time the event will trigger
date	date	Is the actual date the event will trigger like "3/4/99"
m	boolean	TRUE = include Monday
t	boolean	TRUE = include Tuesday
w	boolean	TRUE = include Wednesday
th	boolean	TRUE = include Thursday
f	boolean	TRUE = include Friday
sa	boolean	TRUE = include Saturday
su	boolean	TRUE = include Sunday
devices	string	<p>Is a string with a list of device-command combinations like "B2:on,b3:dimto-50". Valid commands are any X10 command along with the following:</p> <p><i>dimto</i> -&gt; dims the device to the specified value using the current dim level. If the device was at 20% and the command "B2:dimto-50" was given, the actual X10 command send would be "B2 bright 30%"</p> <p><i>dim</i> or <i>bright</i> -&gt; Will first send a bright 100% to the device then dim to the specified level.</p> <p><i>dimr</i> -&gt; Will do a relative dim and simply send the dim value. If the command was "B2:dimr-30", the X10 command "B2 dim 30" would be sent.</p> <p><i>brightr</i> -&gt; The same as "dimr" except the X10 bright command is used.</p>
DeleteWhenTriggered	boolean	Is either TRUE for delete the trigger after it triggers or a FALSE to not delete it.
scr	string	Is the name of the script to run.
group	string (optional)	The group the event will appear in. This can be a new group. This is optional and may be omitted.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	boolean	TRUE if successful else FALSE if error.

Creates a new event with the trigger set to a specific time.

#### Example

The following example creates a new event named "test event" that is triggered at 4:00 pm everyday and dims the device "A1" to the 60% level and does not delete the event after it's triggered.:

```
hs.NewTimeEvent "test event","4:00 pm", "2/2/99", 1, 1, 1, 1, 1, 1, 1, 1, "A1:dimto-60", 0, "", "my group"
```

### 15.8.7.7 NewRecurringEvent

Parameter	Type	Description
evname	string	Is the name that will be given to the new event.
min	interger	Is the time interval in minutes the event will trigger (1 = every minute)
m	boolean	TRUE = include Monday
t	boolean	TRUE = include Tuesday
w	boolean	TRUE = include Wednesday
th	boolean	TRUE = include Thrusday
f	boolean	TRUE = include Friday
sa	boolean	TRUE = include Saturday
su	boolean	TRUE = include Sunday
devices	string	Is a string with a list of device-command combinations like "B2:on,b3:dimto-50". Valid commands are any X10 command along with the following:  <i>dimto</i> -> dims the device to the specified value using the current dim level. If the device was at 20% and the command "B2:dimto-50" was given, the actual X10 command send would be "B2 bright 30%"  <i>dim</i> or <i>bright</i> -> Will first send a bright 100% to the device then dim to the specified level.  <i>dimr</i> -> Will do a relative dim and simply send the dim value. If the command was "B2:dimr-30", the X10 command "B2 dim 30" would be sent.  <i>brightr</i> -> The same as "dimr" except the X10 bright command is used.
DeleteWhenTriggered	boolean	Is either TRUE for delete the trigger after it triggers or a FALSE to not delete it.
scr	string	Is the name of the script to run.
group	string (optional)	The group the event will appear in. This can be a new group. This is optional and may be omitted.
Return value	Type	Description
	boolean	TRUE if successful else FALSE if error.

Creates a new event with the trigger set to a specific time.

#### Example

The following would create an event name "newone" and run the script "check.txt" every

minute only on Monday:

```
hs.NewRecurringEvent "newone",1,1,0,0,0,0,0,0,"",0,"check.txt","my group"
```

### 15.8.7.8 DelayTrigger

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
secs	long	Is the number of seconds before the event name evname is triggered
evname	string	Is the text name of the event that will be triggered.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Trigger the given event after the given number of seconds have elapsed. This is handy if you would like to turn on or off some device a few seconds after the initial event triggers. Note that you can call this as many times as you like, as the events to trigger are held in a queue.

#### Example

```
sub main()
' delay the execution of the event named "lights off" by 5 minutes
hs.DelayTrigger 300, "lights off"
end sub
```

### 15.8.7.9 EventCount

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	integer	

Returns the total number of events configured in the system. Use this with the "GetEvent()" function to iterate through all the events and retrieve their properties.

#### See also

GetEvent

### 15.8.7.10 GetEvent

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Index of desired event

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	EventClass	

Returns an event. The event is of type "EventClass" and can be used to retrieve and set properties of an event. The following code will iterate through all events and display their names.

```
sub main()
  dim ev
  dim ev_count
  dim i
  dim s

  ev_count = hs.EventCount
  for i=1 to ev_count
    set ev=hs.GetEvent(i)
    s=s+ev.name+vbnewline
  next
  msgbox s
end sub
```

An EventClass has a number of properties that holds information about an event. You can access these properties to get and set this information. The function "NewEvent()" can be used to create a new empty event. The properties of the event are listed below.

### EventClass Object

Property	Type	Description
<b>name</b>	string	The name given to the event
<b>ev_time</b>	string	Specifies the absolute time the event will trigger. Only valid for event type TYPE_TIME.
<b>vcmd</b>	string	The voice command for the event
<b>ev_abs_time</b>	integer	The type of event. The following types are defined. <b>time</b> = 0, The event triggers at an absolute time. The property <i>ev_time</i> must contain a valid time. <b>sunrise</b> = 1, The event triggers at sunrise. The property <i>ev_offset</i> may be used to specify an offset from sunrise. Before or after is specified in <i>ev_offset_before</i> . <b>sunset</b> = 2, The event triggers at sunset. The property <i>ev_offset</i> may be used to specify an offset from sunset. Before or after is specified in <i>ev_offset_before</i> . <b>X10</b> = 3, The event triggers on reception of an X10 command. The property <i>ev_trig_hc</i> must contain the X10 housecode, <i>ev_trig_dc</i> must contain the X10 devicecode, and <i>ev_trig_func</i> must contain the X10 command. <b>condition</b> = 4, The event triggers when the given condition is true. The property <i>ev_conditions</i> must contain at least one condition. <b>email</b> = 5, The event triggers on reception of email, either from a POP mail account or MAPI. The property <i>pop_from</i> must contain the email address to check for. <b>recurring</b> = 6, The event is a recurring event and recurs at the interval set in the property <i>rec_secs</i> . <b>manual</b> = 7, The event can only be triggered manually by the web page or menu. If a voice command is set for the event, voice commands will still trigger the event as long as its enabled. <b>irmatch</b> = 8, The event triggers by a match on infrared received. The "irmatch" property holds the infrared number to match. <b>security</b> = 9, security panel event, only the Napco supports this event. <b>phone</b> = 10, The event triggers on a match from a phone event (HomeSeer Phone). The action parameters are stored in the

		<p>ev.phone_actions property. The trigger is held in:  ev.phone_line = phone line to trigger on (1 or 2)  ev.phone_event = phone event number. Values are:  0 = On first ring from outside  1 = On each ring from outside  2 = On first ring from inside  3 = On each ring from inside  4 = On Caller ID available  5 = On answer outside  6 = On answer inside  7 = On message left  8 = On message read  9 = Handset Off Hook  10 = Handset On Hook  * on/off hook events are only supported with hardware that can generate these events</p> <p><b>status change</b> = 11, The event triggers on the change of status of a device. Parameters are:  ev_trig_hc = reference # of device that will change status  ev_trig_dc = X10 command that device must change to  value change = 12, The event triggers on a change of value of a device. Parameters are:  ev_trig_hc = reference # of device that will change status  ev_trig_dc = X10 command that device must change to  ev_time = value to match</p>
<b>phone_actions</b>	string	<p>Holds the actions for a phone trigger. The actions are stored as a string separated with a ":" in the format:  change rings enable (0 or 1) [:] number of rings to change to [:]  phone line number [:] change greeting enable (0 or 1) [:] new greeting</p>
<b>ev_app_path</b>	string	<p>A path to an application that will be launched when the event is triggered. Set to an empty string to be ignored.</p>
<b>ev_date</b>	string	<p>The date the event will trigger. Only valid for "TYPE_TIME" events.</p>
<b>ev_days</b>	integer	<p>A bitmap specifying which days the event may trigger. Only valid for "TYPE_TIME" events. The bits are as follows:  DAY_MON = &amp;H1  DAY_TUE = &amp;H2  DAY_WED = &amp;H4  DAY_THU = &amp;H8  DAY_FRI = &amp;H10  DAY_SAT = &amp;H20  DAY_SUN = &amp;H40</p>
<b>ev_func</b>	integer	<p>The global X10 command to execute when the event triggers. This is used for "ALL_LIGHTS_ON, ALL_UNITS_OFF" commands. Set to "NO_X10" or 17 to do nothing. The property ev_hc must be set to the proper housecode.</p>
<b>ev_hc</b>	integer	<p>Used with ev_func to send global X10 commands (housecode wide). Set this property to housecode you wish to send the command on.</p>
<b>ev_mail_message</b>	string	<p>The message body for email notifications. Only valid if event action is set to "MISC_SEND_EMAIL". See ev_misc.</p>
<b>ev_mail_subj</b>	string	<p>The subject of the message for email notifications. Only valid if event action is set to "MISC_SEND_EMAIL". See ev_misc.</p>
<b>ev_offset</b>	integer	<p>Used with trigger type TYPE_SUNRISE and TYPE_SUNSET. A positive value that is either added or subtracted from sunrise or</p>



		sunset depending on the value of <code>ev_offset_before</code> .
<b>ev_offset_before</b>	boolean	If true, <code>ev_offset</code> is subtracted from sunrise/sunset to get the trigger time, if false, <code>ev_offset</code> is added to sunrise/sunset to get the trigger time.
<b>ev_trig_hc</b>	string	The housecode that will trigger the event when the trigger type is <code>TYPE_X10</code> .
<b>ev_trig_dc</b>	string	The devicecode that will trigger the event when the trigger type is <code>TYPE_X10</code> . A value of "17" may be set to indicate ANY device on the given housecode.
<b>ev_trig_func</b>	integer	The X10 command that will cause the event to trigger. Used with <code>ev_trig_hc</code> and <code>ev_trig_dc</code> and trigger type <code>TYPE_X10</code> .
<b>ev_wav_path</b>	string	The path to a ".wav" sound file that will be played with the event triggers. Set to an empty string to be ignored.
<b>events</b>	string	A list of events to trigger. The list is a comma-separated list of event names. Set to an empty string to be ignored.
<b>ir</b>		The ir commands to be sent to the infrared device.
<b>mail_attach_path</b>	string	A path to a file to attach to outgoing email notifications. Set to an empty string for no attachments.
<b>misc</b>	long	<p>Various control flags as bits in the long value. Bits are defined as:</p> <pre> Public Const MISC_SEND_EMAIL = 1 send an email notification Public Const MISC_SMART_ON = 2 only send ON/DIM/BRIGHT if X10 device is OFF Public Const MISC_ALL_MAIL = 4 all received mail causes trigger Public Const MISC_VOICE_COMMAND = 8 voice command is enabled Public Const MISC_DEL_AFTER_TRIG = &amp;H10 delete the event after it triggers Public Const MISC_DISABLED = &amp;H20 1=event is disabled Public Const MISC_HIDE = &amp;H40 1=this event does not show in views Public Const MISC_APPLY_COND = &amp;H80 1=apply conditions settings to trigger Public Const MISC_ANY_X10 = &amp;H100 1=any x10 command caused trigger if trig type is x10 Public Const MISC_SMART_GROUP = &amp;H200 1=groups devices together when sending X10 commands Public Const MISC_REC_REF_HOUR = &amp;H400 1=recurring minutes are referenced from the hour (recurring trigger type only) Public Const MISC_GUESTS_CAN_TRIGGER = &amp;H800 1=users who login to the web site as guests can trigger this event Public Const MISC_SECURITY = &amp;H1000 1 = vary trigger time by +-30 minutes randomly Public Const MISC_NO_RETRIGGER = &amp;H2000 1 = event cannot retrigger within minute (conditions only) Public Const MISC_EVNO_LOG = &amp;H4000 1= do not log for this event </pre>
<b>misc2</b>	long	More various flags to control actions. They are:

		<pre>Public Const MISC2_VOICE_PHONE_COMMAND = 1 1 = voice command works over the phone Public Const MISC2_VOICE_CONFIRM = 2 1 = voice command in event must be confirmed Public Const MISC2_CHECK_CONDITIONS = 4 1 = when triggering other events, apply conditions Public Const MISC2_SCRIPT_SINGLE_INSTANCE = 8 1 = only one copy of the script may run when the event triggers Public Const MISC2_DELAYED_ACTION = &amp;H10 1 = set if the event was dynamically created due to a delayed action Public Const MISC2_REC_REF_HOUR_ONCE = &amp;H20 1 = for recurring events, only trigger once after the hour Public Const MISC2_SPEAK_ASYNC = &amp;H40 1 = speak TTS in the background Public Const MISC2_INCLUDE_PF = &amp;H80 1 = include this event in power failure recovery</pre>
<b>popfrom</b>	string	Email address that must match for event to trigger. Only valid if trigger mode is TYPE_POP_MAIL.
<b>rec_secs</b>	integer	Number of seconds between triggers for events of trigger type TYPE_RECURRING.
<b>scripts</b>	string	A list of scripts that will run when the event triggers. The list is a comma-separated list of script names. Set this to an empty string for no scripts.
<b>speech</b>	string	The string to be spoken when the event triggers. Set to an empty string for no speech.
<b>security_offset</b>	integer	An offset that is added to the given trigger time. This offset is randomly generated for security mode.
<b>ev_conditions</b>	collection	<p>This is a collection of "condition classes". Each entry defines a condition that must be true before the event will trigger. All conditions must be true. The trigger type must be set to TYPE_CONDITION. See NewCondition for information on creating new entries. The condition class has the following properties:</p> <p><b>ctype (integer)</b> There are two types of conditions, time based and device based. This property defines the type.</p> <pre>Public Const COND_TYPE_TIME = 0 Public Const COND_TYPE_DEVICE = 1</pre> <p><b>condition (integer)</b> For each type above, there are various conditions. For time conditions, this property can be one of the following:</p> <pre>COND_BEFORE_SUNSET = 0 COND_AFTER_SUNSET = 1 COND_BEFORE_SUNRISE = 2 COND_AFTER_SUNRISE = 3 COND_BEFORE_SUNSET_OFFSET = 4 COND_AFTER_SUNSET_OFFSET = 5 COND_BEFORE_SUNRISE_OFFSET = 6 COND_AFTER_SUNRISE_OFFSET = 7 COND_AT = 8 COND_BEFORE = 9           ' before a time COND_AFTER = 10          ' after a time COND_DAY = 11 COND_NIGHT = 12</pre>

		<p>For device conditions, it must be one of the following:</p> <pre> COND_IS_ON = 0 COND_IS_OFF = 1 COND_HAS_BEEN_ON = 2 COND_HAS_BEEN_OFF = 3 COND_X10_ON = 4 COND_X10_OFF = 5 COND_X10_DIM = 6 COND_X10_BRIGHT = 7 COND_HAS_BEEN_ON_MORE = 8 COND_HAS_BEEN_OFF_MORE = 9 COND_X10_ANY = 10 COND_VALUE_EQUALS = 11 </pre> <p><b>dev_ref (single)</b> For device conditions, the condition refers to a particular device. Each device has a unique reference number. You can use the device functions to find the particular device, get its reference number, and load it here in this property.</p>
<b>time</b>	string	For time based conditions, this property holds time values as a sting like: 4:20 pm. Hours and minutes of duration are set to values like: 2:30
<b>ev_devices</b>	collection	<p>This is a collection of classes of type "devfunc". This is a list of devices that the event controls. See NewDevfunc and NewCondition for information on creating new entries. The devfunc class has the following properties:</p> <p><b>devid (single)</b> This is the reference id for the device to control. Each device has a unique ID that can be obtained with the device functions.</p> <p><b>dim_percent (integer)</b> The dim value to dim or bright the device to. The range is 0-100.</p> <p><b>dimto (integer)</b> 0=relative 1=Absolute 2=Absolute bright to 100% first</p> <p><b>func (integer)</b> The X10 function to send to the device.</p> <p><b>secs_delay (long)</b> The time in seconds to delay before sending the requested command to the device.</p> <p><b>new_string (string)</b> The string to set the device to when the action is performed. The new string will be displayed as the device's status.</p>
<b>group</b>	string	The name of the group the event belongs to

### Example

The following script shows how to reiterate though all events, get the event name, and reiterate though all the devices the event controls.

```

sub main()
    dim ev          ' eventclass class
    dim ev_count
    dim i
    dim s
    dim dv          ' devfunc class

    ev_count = hs.EventCount
    set ev=hs.GetEvent(1)
    ' ev.name="MY NEW NAME"          ' this will change the value
    for i=1 to ev_count
        set ev=hs.GetEvent(i)
        s=s+ev.name+vbnewline
        for each dv in ev.ev_devices
            s=s+"++"+get_device_name(dv.devid)+vbnewline
        next
    next
    msgbox s

end sub

function get_device_name(ref)
    dim d
    dim count
    dim i
    dim s

    count = hs.DeviceCount
    s=cstr(ref)+"="
    for i=1 to count
        set d = hs.GetDevice(i)
        if ref = d.ref then
            get_device_name = d.name
            exit function
        end if
    next
end function

```

**See also**

EventCount

**15.8.7.11 GetEventEx**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
event name	string	Name of event

<u>Return value</u>	<u>Type</u>	<u>Description</u>
event reference	object as EventClass	Returns a reference to an event with the given name. If multiple events have the same name, the wrong event may be returned. The returned object can be used to access properties of the event

**Example**

```

dim ev
set ev = hs.GetEventEx("night event")
msgbox "The event type is: " & cstr(ev.ev_abs_time)

```

### 15.8.7.12 **GetEventList**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
event list	collection	Returns a reference to all HomeSeer events. HomeSeer stores events in a collection of EventClass objects.

#### Example

```

sub main

dim events
dim ev

set events = hs.GetEventList

' cycle through all events and get their names
for each ev in events
    hs.WriteLog "Event Info", "Event name is: " & cstr(ev.name)
next

end sub

```

### 15.8.7.13 **EventExists**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Name of the event. The name is not case sensitive.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
event index	Boolean	returns TRUE if the given event exists, else returns false

Checks if a given events exists in HomeSeer's event list.

### 15.8.7.14 **NewEvent**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
name	string	Name of the new event

<u>Return value</u>	<u>Type</u>	<u>Description</u>
event index	integer	index number of the new event

Creates a new empty event. The trigger mode is set to MANUAL. All other properties are cleared, the name is set to the name given. The index of the new event is returned and may be used in the "GetEvent" call to set properties of the new event.

#### Example

This example creates a new event and then gets the event name.

```

sub new_event(name)
dim i
dim ev

```

```

i=hs.NewEvent(name)
set ev=hs.GetEvent(i)
msgbox ev.name
end sub

```

**See also**

GetEvent  
EventCount

**15.8.7.15 NewCondition**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
condition reference	object type condition	returns a reference to the new condition object

Creates a new condition class for use with GetEvent. This new condition can then be added to an event that uses conditions as the trigger.

**Example**

This example creates a new event, sets the event type to *Condition*, and then adds a condition that triggers at 3:00 PM.

```

sub main()

dim ev
dim index
dim cd

index = hs.NewEvent("my_condition")
set ev = hs.GetEvent(index)

ev.ev_abs_time = 4
set cd = hs.NewCondition
cd.ctype = 0
cd.condition = 8
cd.time = "3:00 PM"
ev.ev_conditions.add cd

end sub

```

**See also**

GetEvent

### 15.8.7.16 RemoveDelayedEvent

Parameter	Type	Description
device	string	Is the X10 code of the device like "A1"
event_name	string	Is the name of the event that is to be removed from the queue.
Return value	Type	Description
none		

Removes a previously queued event from the pending event queue. There are two types of pending events, the first is an X10 operation and the pending event contains only the housecode and unit code of the X10 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.

#### See also

DelayTrigger

### 15.8.7.17 NewDevFunc

Parameter	Type	Description
none		
Return value	Type	Description
devfunc	object	Reference to a device function class.

Creates a new class of type devfunc. This class is used to describe the function of an X10 device in an event. There is one function for each X10 device in the event.

The attributes of this class are:

#### devid

The reference id of the device. Reference id's are stored as the property ".ref" with the actual device. The id can be obtained using GetDevice. The id must be set for the device to be referenced properly from an event.

#### func

The X10 command to be sent to the device. These are standard X10 codes like 2 for ON, 3 for OFF, etc. If the device is a thermostat, this property describes the following settings:

```
0=heat 1=cool 2=auto 3=do not set
```

#### dimto

The type of dimming to be performed on the module if the function is dim or bright. The settings are:

```
DIMTO_RELATIVE = 0
DIMTO_ABSOLUTE = 1
DIMTO_ABSOLUTE_BRIGHT_FIRST = 2
```

If the device is a thermostat, this holds the actions for the fan. The settings are:

```
0=on 1=off 2=auto 3=do not set
```

**dim\_percent**

The percentage to dim or brighten a device. If the device is a thermostat, this property holds the temperature to set the setpoint to. If set to -99, no setpoint is set.

**secs\_delay**

Holds the number of seconds to delay before executing the device actions.

**new\_string**

The string to set in the device. The new string will be displayed as the device status.

**Example**

The following example creates a single device, then creates an event and adds the new device to it.

```
sub CreateDeviceEvent()  
    dim id  
    dim i  
    dim dev  
    dim evl  
    dim df  
  
    ' create a new empty device  
    i=hs.NewDevice("TestDevice")  
    set dev = hs.GetDevice(i)  
  
    ' set some attributes of the device  
    dev.location = "kitchen"  
    dev.name = "lamp"  
    dev.hc = "A"  
    dev.dc = "16"  
  
    ' get the id for later  
    id = dev.ref  
  
    ' create a new empty event  
    i=hs.NewEvent("test_name")  
    set evl = hs.GetEvent(i)  
  
    ' create a new devfunc class to describe a device  
    set df=hs.NewDevFunc  
  
    ' add a reference to the new device  
    df.devid = id  
  
    ' set the actions for this device  
    df.func = 2      ' ON  
  
    ' add the class to our collection in the event  
    evl.ev_devices.add df  
  
end sub
```

**See also**

GetEvent  
GetDevice



### 15.8.7.18 SaveEventsDevices

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

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 Windows user interface.

## 15.8.8 EMAIL Functions

### 15.8.8.1 SendEmail

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
mtto	string	Is the address you are sending the mail to.
mfrom	string	Is the address you are sending from. Note that some ISP will not allow you to put just anything in this field. You may be required to put your real email address here. If you are using MAPI to handle your email, MAPI will enter your email address that is associated with your default email account. This field will be ignored.
msubject	string	Is the subject of the mail.
message	string	Is the body of the mail.
attach	string (optional)	An absolute pathname to a file to attach to the email.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

This will send an email message.

The attach parameter is useful if you would like to email a picture taken with an attached digital camera. Just attach the path to any picture file created by the camera.

### 15.8.8.2 MailMsgCount

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
count	integer	Returns the total number of unread messages that are in your inbox. This will only work if MAPI is enabled as your email interface. (menu <i>View-&gt;Options-&gt;email</i> )

### 15.8.8.3 MailFrom

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be retrieved.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
from address	string	The email address of the sender for the indexed message.

### 15.8.8.4 MailFromDisplay

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be retrieved.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
from name	string	The name of the sender for the indexed message.

### 15.8.8.5 MailDate

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be retrieved.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	The date of the indexed mail message.

### 15.8.8.6 MailSubject

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be retrieved.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	The date of the indexed mail message.

### 15.8.8.7 MailText

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be retrieved.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	A string that is the body of the indexed email message.

### 15.8.8.8 MailTo

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be retrieved.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	A string that is the address of who the message was sent to.

**15.8.8.9 MailToDisplay**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be retrieved.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	string	A string that is the name of the person the message was sent to.

**15.8.8.10 MailDelete**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
index	integer	Is the index number of the message to be deleted.
<u>Return value</u>	<u>Type</u>	<u>Description</u>

**15.8.8.11 MailTrigger**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
none		
<u>Return value</u>	<u>Type</u>	<u>Description</u>
	long	The index of the email message that was last received.

Returns the index into the MAPI message list of the message that caused the last trigger. If you create an event that triggers when an email is received, 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.

**Example**

```
' this script will access the email 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
```

## 15.8.9 CPU-XA/Ocelot Functions

### 15.8.9.1 `cpuxa.GetPoint`

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
unit	integer	The unit number of the target device. 1 = the first unit.
point	integer	The input point of the target device. 0 = the first input point.
digital	boolean	TRUE = get a digital input reading, FALSE = get an analog input reading. The units point must be setup with a jumper to either analog or digital. A false reading will be returned if the point is set for digital, but is read as analog.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
I/O point	integer	see below

This function is part of the `cpuxa` object. It will return the value of a specific input point on a `secu16` module. If the `digital` parameter is `TRUE`, the result will be either 0 for a low input or 1 for a high input. If `digital` is false, the result will be a value between 0 and 255, which represents the voltage level. The `secu16` needs to have the proper jumpers set for each input. Refer to the CPU-XA/Ocelot users guide for more information.

#### **Example**

Read the digital input from the first point in the first unit of the `SECU16`.

```
sub main()
dim v
v=cpuxa.GetPoint(1,0,TRUE)
end sub
```

#### **See Also**

`cpuxa.SetPoint`

### 15.8.9.2 `cpuxa.LearnIRStart`

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
location	integer	The location number in the CPU-XA where the IR signal will be stored. Some units can hold 512 commands and others can hold 1024. The first IR location is 0.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Member of the `cpuxa` object.

When learning infrared from the CPU-XA unit, this will be called first. This puts the CPU-XA into learn mode and it awaits an IR signal at its input window. *LearnIREnd* should then be called to wait for the learn to complete. When *LearnIREnd* returns, the signal has been learned.

### Example

This example will learn an IR signal into location 0.

```
sub main()

' start the learn
cpuxa.LearnIRStart 0

' wait to complete
cpuxa.LearnIREnd

' signal to be learned should be generated at the units input window at
this time

end sub
```

### See Also

cpuxa.LearnIREnd

#### 15.8.9.3 cpuxa.LearnIREnd

Parameter	Type	Description
none		

Return value	Type	Description
none		

Member of the cpuxa object.

Called after LearnIRStart to wait for the infrared learn to complete. See LearnIRStart for more information and an example.

### See Also

cpuxa.LearnIRStart

#### 15.8.9.4 cpuxa.PlayIR

Parameter	Type	Description
location	integer	The location number in the CPU-XA where the IR signal will be stored. Some units can hold 512 commands and others can hold 1024. The first IR location is 0.

Return value	Type	Description
none		

Member of the cpuxa object.

Informs the CPU-XA to send a previously learned infrared command.

### See Also

cpuxa.LearnIRStart  
cpuxa.LearnIREnd

### 15.8.9.5 cpuxa.SendRaw

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
location	integer	The location number in the CPU-XA where the IR signal will be stored. Some units can hold 512 commands and others can hold 1024. The first IR location is 0.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Member of the cpuxa object

Sends a raw buffer of data to the CPU-XA . Refer to the Applied Digital developer's documentation for information on how to format the buffer.

### 15.8.9.6 cpuxa.SetVar

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
vnum	integer	The variable number to set.
data	integer	The value to set the variable to.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Member of the cpuxa object.

Sets a specified variable in the CPU-XA.

### 15.8.9.7 cpuxa.SetPoint

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
unit	integer	The unit number of the target device. 1 = the first unit.
point	integer	The output point of the target device. 0 = the first output point.
value	integer	The value to set, either 1 for ON or 0 for OFF.
<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Member of the cpuxa object.

Sets an output relay to either ON or OFF in the SECU16.

### 15.8.9.8 **cpuxa.PlayRemoteIR**

<u>Parameter</u>	<u>Type</u>	<u>Description</u>
unit	integer	The unit number of the SECUIR. 1 = the first unit.
zone	integer	The zone to send the IR on.
location	integer	The location number of the IR command to send.

<u>Return value</u>	<u>Type</u>	<u>Description</u>
none		

Member of the cpuxa object.

Transmits an infrared signal from a remote SECUIR module.



### 15.8.10 System Functions

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). See the function descriptions in the individual help files for more information about these functions.

#### Speak a phrase

```
system.speak txt 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 commands can be re-enabled.

```
system.ClearAllVoiceCommands
```

#### Stop voice recognition

```
system.StopListen
```

#### Start voice recognition

```
system.StartListen
```

#### Get the phone line that triggered the event (0=triggered by HomeSeer, and not the phone)

```
system.LastLine or system.SpeakSource
```

#### Start voice recognition

```
system.StartListen
```

# Appendix A

## HomeSeer as an ActiveX Server

You can control HomeSeer from your own applications. Simply add a reference to it from within your software. You will have access to all of the properties and methods described in the Scripting Interface section.

You can call into HomeSeer and register a form object that will be called when HomeSeer receives an X10 command. Call the following function:

```
hs.RegisterEventCB evtype as long, obj as object
```

The *evtype* parameter is a bitmap of event types. The event types are defined as follows:

```
const ev_type_x10 = 1           ' X10 received event
const ev_type_log = 2          ' a message written to the event log
const ev_type_status_change = 4 ' a device changed status
const ev_type_audio = 5
```

The *obj* parameter should be a reference to a form with the following method declared public:

```
Public Sub HSEvent(parms as variant)
End Sub
```

The parameters are passed in a variant that is an array of variants. Each entry in the array is a parameter. The number of entries depends on the type of event and are described below.

The event type is always present in the first entry or *parms(0)*.

The *parms(0)* event type is defined as follows:

```
1 = X10 event
2 = Event log message event
4 = device status change
5 = audio start/stop
```

Parameters for the X10 event:

```
parms(1)=devices = [device string like "A1+2"]
parms(2)=housecode = [the houscode the command was heard on like "A"]
parms(3)=command = [the actual X10 command, in the range 1 -> 16]
parms(4)=dim_level = [normally the dim level from a dim command, or data byte 1
                    for the extended X10 command]
parms(5)=data2 = [data byte 2 for the extended X10 command]
```

Parameters for the event log event are:

```
parms(1)=date_time = [date and time of log entry]
parms(2)=msg_class = [type of log entry, ie: "speak" or "ERROR", etc]
parms(3)=msg = [the actual event log message]
```

Parameters for the status change event are:

```
parms(1)=device = [device that changed status]
parms(2)=housecode = [housecode of device that changed status]
parms(3)=command = [command sent to device (X10 command)]
```

```
parms(4)=extra = [first byte of extended data, like dim level]
parms(5)=data2 = [second byte of Extended X10 command]
```

Parameters for the audio event are:

```
parms(1)=started = [true=audio channel is now being used (TTS or WAV playing)
False = device is now free]
parms(2)=deviceId = [device ID of audio device that is being used]
```

*Other events may be added in the future.*

HomeSeer will then call the *HSEvent* method when the requested event is available. Note that this method should return as quickly as possible as to not interfere with normal HomeSeer event processing.

Here is some sample code for a client using the HomeSeer server. Add a reference to the "HomeSeer" object to your project.

```
Dim hs As HomeSeer.Application

Const EV_TYPE_X10 = 1
Const EV_TYPE_LOG = 2

' speak some text
Private Sub Command1_Click()
    hs.Speak "hello"
End Sub

' force an event callback
Private Sub Command2_Click()
    hs.SetX10Event "B1", "B", 2, "", ""
End Sub

' get a reference to the HomeSeer ActiveX interface and register
' to receive the X10 and event log events (bits 1 and 2 are set in
' in the evtype parameter
Private Sub Form_Load()
    Set hs = New HomeSeer.Application
    hs.RegisterEventCB 3, Me
End Sub

' this is the callback method that HomeSeer calls when an event is available
Public Sub HSEvent(parms As Variant)
    Debug.Print "HS_EVENT"

    Select Case parms(0)
        Case EV_TYPE_X10
            Text1 = Text1 + "HS EVENT: " + Str(parms(0)) + " Dev:" + parms(1)
+ " HC: " + parms(2) + " CMD: " + Str(parms(3)) + vbCrLf
            Case EV_TYPE_LOG
                Text1 = Text1 + "HS EVENT: " + Str(parms(0)) + " Date/Time:" +
parms(1) + " Type: " + parms(2) + " Msg: " + parms(3) + vbCrLf
            End Select
    End Sub

' Get the status of a device
Private Sub status_Click()
    Dim i As Long
    i = hs.DeviceStatus("B2")
    Text1 = Str(i)
End Sub
```

```
'Send an X10 command
Private Sub x10_Click()
    hs.ExecX10 "B2", "on", 50
End Sub
```

```
'Speak
Private Sub speak_Click()
    hs.Speak "hello"
End Sub
```

## 16 Index

ActiveX server		Running HomeSeer for the first time	14
using HomeSeer as	209	Scripting	
Assistant Wizard	15	adding to event	49
autosave	10	overview	103
BETA	14	reading and writing files	105
check_vcnd.txt	58	statements	50
conditions		Scripting Functions	
applying to a trigger	35	AddVoiceCommand	146, 147
Conditions		ClearAllVoiceCommands	108, 146, 148
How they work	41	ClearLastX10	109, 178
Devices		CloseComPort	108, 142, 143
Creating/Editing	23	cpuxa.GetPoint	112, 204
custom types	29	cpuxa.LearnIREnd	205
deleting	29	cpuxa.LearnIRStart	112, 204, 206
supported types	23	cpuxa.PlayIR	112, 205
email		cpuxa.PlayRemotelR	112, 207
controlling devices by	21	cpuxa.SendRaw	112, 206
setting up	15	cpuxa.SetPoint	112, 204, 206
Event Log	18	cpuxa.SetVar	112, 206
Events		CreateVar	113, 114
action order	44	DelayTrigger	111, 189, 198
all unit/lights control	47	DeleteEvent	186
as voice command	32	DeleteVar	113, 114
controlling multiple devices	31	DeviceCount	109, 158
controlling thermostats with	46	DeviceStatus	109, 163, 164, 165, 166, 167
create for a single device	31	DeviceString	109, 155, 170
disable logging	35	DeviceStringByName	109, 155
launching an application	48	DeviceTime	109, 174, 176, 177
overview	31	DeviceTimeByName	109, 179
playing a sound	48	DeviceValue	109, 167, 169, 172
running a script	49	DeviceValueByName	109, 170, 172, 173
security panel actions	53	DisableEvent	111, 186
security panel trigger	53	EnableEvent	111, 186
security timing	35	EventCount	111, 189, 195, 197
sending email	48	ExecX10	110, 161
sending infrared commands	52	ExecX10ByName	110, 160
speaking with text-to-speech	49	GetAppPath	106, 120, 131
trigger types	33	GetComPortCount	108, 142, 143
triggering other events	52	GetComPortData	108, 142, 143
Events View	18	GetDevice	110, 155, 156, 158, 199
Features	12	GetDeviceCode	110, 159
Groups	54	GetEvent	111, 189, 195, 196, 197, 199
Infrared		GetIPAddress	106, 121
matching on reception	53	GetLastEvent	111, 185
Infrared control		GetURL	106, 116, 117
delaying sequences	53	GetVar	106, 113, 114
devices supported	63, 69	IsOff	162, 163, 164, 165, 167
learning	65	IsOffByName	110, 162, 163, 164, 165, 167
overview	63, 69	IsOn	162, 163, 164, 165, 167
JDS IRxpander	12	IsOnByName	110, 162, 163, 164, 165, 167
Leviton 6381	26	Keys	106, 121
MAPI	15	LastCommandSelected	109, 147
NAPCO Gemini	12	LastVoiceCommand	108, 146
Options Summary	97	LastX10	110, 177
PCRemote	12	ListenForCommands	109, 149
Printing	93	MailDate	111, 202
Quick Start	11	MailFrom	111, 201

MailMsgCount.....	111, 201	random times .....	54
MailSubject .....	111, 202	SmartGroup .....	46
MailText .....	112, 202, 203	SmartLinc SwitchLinc .....	26
MailTrigger .....	112, 203	SmartOn	
NewCondition .....	111, 197, 200	event view .....	18
NewDevFunc .....	111, 198	resetting device timers .....	41
NewDevice.....	110, 155, 156	SmartON	
NewEvent .....	111, 196	how to use .....	46
NewRecurringEvent.....	188	SMTP .....	16
NewTimeEvent .....	111, 187	status display.....	28
OpenComPort.....	108, 141, 143, 144	Sunrise/Sunset	
PollDevice.....	159	setting up .....	15
RemoveDelayedEvent .....	111, 198	System Requirements .....	13
Run .....	107, 124, 126	Technical Support .....	13
RunEx.....	107, 124	Thermostat	
SaveVar .....	107, 113, 114	setting up .....	87
SendEmail .....	112, 201	Timers .....	53
SendIR.....	108, 145	Toolbar .....	20
SendToComPort .....	108, 142, 144	Trigger	
SetDeviceStatus .....	110, 163, 164, 165, 166, 167	apply conditions to .....	38
SetDeviceString .....	110, 155, 170	by absolute time.....	36
SetDeviceStringByName .....	110, 155, 170	by condtion .....	40
SetDeviceValue .....	110, 168, 169, 173	by email reception.....	43
SetDeviceValueByName .....	173	by sunrise/sunset.....	39
SetSecurityMode .....	107, 127	by web page guests.....	35
SetSpeaker.....	109, 149	by X10 command.....	38
SetVolume .....	107, 115, 136, 137, 181, 183	TV/Video Game Timer.....	95
speak .....	109, 150, 151, 153, 154	Voice Commands	
StartListen .....	109, 149, 150	adding to events .....	34
StopListen.....	149, 150, 154	formatting.....	59
StringItem .....	107, 126	using to control I/R devices.....	67
Sunrise .....	107, 126	Voice Recognition	
Sunset .....	107, 126	without the Agent.....	56
TriggerEvent .....	111, 185	Web Server	
version .....	107, 116	controlling devices from your own pages.....	77
WaitEvents .....	107, 116, 127	creating your own home page .....	77
WaitSecs .....	107, 116, 128	customizing pages .....	80
WriteLog .....	107, 115, 127	limiting access .....	80
Security		overview.....	71
panel support.....	85	starting.....	71