HomeSeer HS3 - Software Development Kit

User Manual

Prepared by: HomeSeer
Date: 3/10/2015

HomeSeer HS3 - Software Development Kit

Table of Contents

Home > Document Revisions

Document Revisions

Changes to this SDK document as listed below.

Date	Change
11/19/12	Initial BETA version released for developers
12/12/12	Added device type information for new device based technology API
12/21/12	Added descriptions of the cslPageBuilder functions, updates to device types
1/10/13	• New help files now include base help file and scripting documentation for the developer's edition. • Extensive updates to the Devices section including CAPI and Value/Status-Value/Graphic pairs, and Device_Type informati • Much of the scripting reference for Events is updated, new commands documented: AddDeviceActionToEvent, EventSetTin EventSetRecurringTrigger, Event_Group_Info, Event_Group_Info_All, Event_Info, Event_Info_All, and Event_Info_Group • Some deprecated commands removed. • Technology APIs update for Thermostat API and Music API. • Change made to prevent special characters for plug-in Name properties with the exception of space, dash, and period.
1/16/13	Reworked much of the MusicAPI
1/22/13	• Added ButtonRender_Row/Column to the value/status pairs. • Added CONTROL_POPUP to the device MISC bits - when set, indicates that the device's controls should appear in a pop-u device utility page.
2/5/13	Changed references of Music API to Media API
2/6/13	Added pages for AdditionalDisplayData and ScaleText
2/15/13	Removed ButtonPress and updated the device type information in the scripting API to reference the fact that a device type will no launch a script when the value or string changes. New properties were added to the DeviceClass in the scripting API to hold the name in the script.
3/6/13	Clarified HasConditions
6/21/13	Updated Testing Your Package section
7/11/13	Added DeviceVSP_GetAllStatus which returns all Status/Both value/status pairs so that the status values for a device may be enun Added ControlUse to VSPair and CAPIControl object - this eNum indicates a control use for lighting including On, Off, Dim, and
12/9/13	Added jqListBoxEx
6/10/14	Modifications to the MusicAPI (Technology APIs) and associated enums
9/4/14	Added JSON section
3/10/14	Changed return value for JSON getsetting

See Also

Getting Started
Controlling with JSON
Plugin Initialization
Base Plugin API
Devices
Callbacks
Triggers
Actions
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater
Appendices

Home > Getting Started

Getting Started

Articles in this section

See Also

Home > Getting Started > Introduction

Introduction

Introduction

This SDK is currently version 1.0 for HomeSeer HS3 October 2012

This SDK is new for HomeSeer HS3 and has many changes over the SDK for HomeSeer HS2. If you have been developing plugins for HS2 please read over this documentation so you are familiar with the changes.

The HomeSeer SDK is a powerful tool for developers to create plug-ins for HomeSeer that integrate seemlessly with the application providing a professional approach to extending HomeSeer's capabilities.

Please note that if you decide you wish to sell your plug-in and you want to use HomeSeer's sales resources and licensing mechanism, you will need to contact Mark Colegrove at Sales@HomeSeer.com to register your name, address (for revenue checks), plug-in name and a marketing/sales description.

The plug-in SDK allows you to get notification of HomeSeer events and allows you to add custom triggers, actions, and conditions to events. If you do not need any of these things and only want to interface something to HomeSeer using HomeSeer devices, then stop right here. Scripts can be used instead of a plug-in to provide a simple, device based interface to an external device. Plug-ins should be used only when tight integration with HomeSeer is desired or necessary.

Some of the features of this SDK are also available in the scripting interface of HomeSeer. The main features or capabilities that this SDK can provide include the following:

- Program-to-Program notifications for certain HomeSeer events such as when HomeSeer is writing to the system log file, when a device's configuration is being changed, or when HomeSeer is speaking.
- Ability to add one or more custom trigger/condition pages to the list of triggers used in events.
- Ability to add one or more custom action pages to the list of actions available in events.
- Ability to add (register) one or more web pages to the HomeSeer built-in web pages for HTML interaction with the users.
- Ability to add options and configuration prompts to devices for purposes of supporting the use of devices in a HomeSeer system with your plug-in.
- Ability to change the behavior of HomeSeer based upon information set in the properties of devices and events.
- A reliable licensing mechanism that prevents unauthorized use of your plug-in.

The HomeSeer Plug-In SDK is entirely separate from the marketing and distribution method you choose for your plug-in. If HomeSeer Technologies is used for the distribution of your plug-in however, you will be responsible for maintenance and testing of an installation package for HomeSeer's software updater tool (The Updater).

Function prototypes in **BLACK** are functions that live in a plug-in, function prototypes in **GREEN** are functions that live in the HomeSeer scripting API, function prototypes in **RED** are callback functions used by plugins only (not available to scripts), **gold** are web page functions in the class clsPageBuilder.

See Also

System Requirements
Architecture

System Requirements

The following is required in order to use this SDK and develop plug-in's for HomeSeer HS3:

1. HomeSeer HS3 Installed on a Windows PC, preferably Windows 7 operating system
2. Visual Studio 2010 or later (previous versions could be used, but the sample projects provided will not load)
3. Knowledge of Visual Basic .NET and the .NET runtime
4. If you intend to support HS3 on Linux, a Linux virtual image is available for testing

See Also

Introduction
Architecture

Architecture

HomeSeer HS3 plugins are simply console EXE programs. They can be windows forms EXE's if you find you need to present the user with a form for some reason. However, most of HomeSeer is accessible through the built-in web server so if your plugin uses a form, the form will not be accessible from the web. Also, windows forms applications will probably have issues under Linux. All the samples provided are written in VB.NET, but any .NET language could be used.

Plug-in's communicate with HomeSeer through a simple TCP connection on port 10400. A communication framework called HSCF is used for 2-way communications and the connection remains open as long as the plug-in is connected. If a connection is lost, HomeSeer will attempt to re-connect with the plug-in. The communication framework also does its best to maintain the connection. This framework was chosen over Windows WCF due to its high performance, simplicity, and better compatibility with Linux.

Plug-ins written for HomeSeer that are not Windows hardware dependent should be tested under the Linux version of HomeSeer to ensure they are compatible.

Plug-ins can run on the local system HomeSeer is running on as well as remote systems, even a system on the Internet. A plug-in can accept a few command line parameters such as:

server=IP ADDRESS
Use this parameter if you are starting a plug-in on a remote system. Pass the IP address of the HomeSeer system that you wish to connect to

instance=INSTANCE NAME
Plug-ins can be enabled to support multiple instances. For example, an X10 plugin-in may be configured to control an X-10 interface on Communications port 1. It may be desirable to install a second X10 plug-in on a different COM port. In this case a second instance can be started. This instance is assigned a unique name and is passed to the plug-in when it is started by HomeSeer. A remove plug-in can also be started with a unique instance name.

Plug-ins conform to a specific interface definition. All plugins must implement the IPlugInAPI interface. Other interfaces may be added if additional functionality is requires, such as thermostats or music. See the Base Plugin API section for information about the IPlugInAPI and the Technology APIs section for information about optional API's.

To get started writing your own plug-in it is advised that you read through this documentation and use the provided sample plug-in as a template for your own plug-in.

See Also

Introduction
System Requirements

Home > Controlling with JSON

Controlling with JSON

HomeSeer can be controlled using JSON formatted data. Some notes on this API:

* Parameters are not case sensitive
* most parameters accept "all" as an option to include all items, or if the parameter is not included "all" is assumed

Testing:

A demo server is available for testing the API, use the following URL:

```
https://connected.homeseer.com/JSON?user=demo@homeseer.com&pass=demo100&request=getstatus
```

The following commands are available:

GET requests

URL	Description

	Returns the status of a device in the system or all the devices in
	following format. The following parameters are supported in or
	filter the response:
	
	location1=loc1 (only return the devices that are in the specific
	location1, omit or set to "all" for all devices at this location)
	location2=loc2 (only return the devices that are in the specific
	location2, omit or set to "all" for all devices at this location)
	ref=## (only return the device that matches the specific refere
	this may be a list of reference #'s like 3467,2342,869, omit or
	"all" to return all devices)
	
	All GET requests are terminated with a CRLF.
	
	If no JSON data is expected from the request, the return may b
	"ok", or "error".
	
	This response is for 2 devices, the first is a thermostat tempera
	the second is a light switch:
	
	`{`
	`  "Name":"HomeSeer Devices",`
	`  "Version":"1.0",`
	`  "Devices":[`
	`    {`
	`      "ref":3398,`
	`      "name":"Temperature",`
	`      "location":"Z-Wave",`
	`      "location2":"Node 122",`
	`      "value":82,`
	`      "status":"82 F",`
	`      "device_type_string":"Z-Wave Temperatu`
	`      "last_change":"\/Date(1410193983884)\/`
	`      "relationship":4,`
	`      "hide_from_view":false,`
	`      "associated_devices":[`
	`        3397`
	`      ],`
	`      "device_type":{`
	`        "Device_API":16,`
	`        "Device_API_Description":"Thermosta`
	`        "Device_Type":2,`
	`        "Device_Type_Description":"Thermost`

/JSON?
request=getstatus&ref=##&location1=LOC1&location2=LOC2

```
                Temperature",
                "Device_SubType":1,
                "Device_SubType_Description":"Tempe
             },
             "device_image":""
          },
          {
             "ref":3570,
             "name":"Switch Binary",
             "location":"Z-Wave",
             "location2":"Node 124",
             "value":255,
             "status":"On",
             "device_type_string":"Z-Wave Switch Bi
             "last_change":"\/Date(1410196540597)\/"
             "relationship":4,
             "hide_from_view":false,
             "associated_devices":[
                3566
             ],
             "device_type":{
                "Device_API":4,
                "Device_API_Description":"Plug-In A
                "Device_Type":0,
                "Device_Type_Description":"Plug-In
                0",
                "Device_SubType":37,
                "Device_SubType_Description":""
             },
             "device_image":""
          }
       ]
    }
```

Where:

ref = unique device reference number, used for any subsequent requests such as device control, leave blank or set to "ALL" to g status for all devices
name = the name of the device
location = the location of the device such as "kitchen"
location2 = the second location of the device such as "first flo
value = the current value of the device, a double
status = the current string that represents the status of the de such as "on" or "off"
last_change = the date/time the device last changed status
relationship = 2:root device (other devices may be part of this physical device),3:standalone=this is the only device that repre this physical device,4:child=this device is part of a group of de that represent this physical device
hide_from_view = true/false if true, this device has been set be visible in any user interface
device_type_string = The string the describes this device, se plugin authors on their devices.
associated_devices = a list of device reference #'s that are associated with this device. If the device is a ROOT device, then is child devices, if the device is a child device, then the list will c one device that is the root device.
device_type = Detailed information about the capabilities of the device, used mainly to determine if the device adheres to other such as thermostat, energy, etc. See the Devices section in the S section in the HomeSeer HS3 user documentation for informatio the properties in this section.

Returns control information for a device in the system, or all de
Devices contain "control pairs", one pair for each possible contr
For example, a light that can be turned on and off would contai
control pairs, one for ON and one for OFF. The control pair des
how to control the device. The most important information is th
and the "ControlValue", as those will be needed when the devic
be controlled. The information from this call can be used to bui
user interface that will control the device.

Parameters:

ref=### (where ### is the device reference #, or "all" to return
information for all devices)

```
{
   "ControlPairs":[
      {
         "Do_Update":true,
         "SingleRangeEntry":true,
         "ControlButtonType":0,
         "ControlButtonCustom":"",
         "CCIndex":0,
         "Range":null,
         "Ref":2256,
         "Label":"On Last Level",
         "ControlType":5,
         "ControlLocation":{
            "Row":1,
            "Column":3,
            "ColumnSpan":0
         },
         "ControlLoc_Row":1,
         "ControlLoc_Column":3,
         "ControlLoc_ColumnSpan":0,
         "ControlUse":4,
         "ControlValue":255,
         "ControlString":"",
         "ControlStringList":null,
         "ControlStringSelected":null,
         "ControlFlag":false
      },
      {
         "Do_Update":true,
         "SingleRangeEntry":true,
         "ControlButtonType":0,
         "ControlButtonCustom":"",
         "CCIndex":1,
         "Range":null,
         "Ref":2256,
         "Label":"On",
         "ControlType":5,
         "ControlLocation":{
            "Row":1,
            "Column":2,
            "ColumnSpan":0
         },
         "ControlLoc_Row":1,
         "ControlLoc_Column":2,
         "ControlLoc_ColumnSpan":0,
         "ControlUse":1,
         "ControlValue":99,
         "ControlString":"",
         "ControlStringList":null,
         "ControlStringSelected":null,
```

/JSON?request=getcontrol&ref=##	see below

```
        "ControlFlag":false
     },
     {
        "Do_Update":true,
        "SingleRangeEntry":true,
        "ControlButtonType":0,
        "ControlButtonCustom":"",
        "CCIndex":2,
        "Range":null,
        "Ref":2256,
        "Label":"Off",
        "ControlType":5,
        "ControlLocation":{
           "Row":1,
           "Column":1,
           "ColumnSpan":0
        },
        "ControlLoc_Row":1,
        "ControlLoc_Column":1,
        "ControlLoc_ColumnSpan":0,
        "ControlUse":2,
        "ControlValue":0,
        "ControlString":"",
        "ControlStringList":null,
        "ControlStringSelected":null,
        "ControlFlag":false
     },
     {
        "Do_Update":true,
        "SingleRangeEntry":true,
        "ControlButtonType":0,
        "ControlButtonCustom":"",
        "CCIndex":3,
        "Range":{
           "RangeStart":1,
           "RangeEnd":98,
           "RangeStatusDecimals":0,
           "RangeStatusValueOffset":0,
           "RangeStatusDivisor":0,
           "ScaleReplace":"",
           "HasScale":false,
           "RangeStatusPrefix":"Dim ",
           "RangeStatusSuffix":"%"
        },
        "Ref":2256,
        "Label":"Dim (value)%",
        "ControlType":7,
        "ControlLocation":{
           "Row":2,
           "Column":1,
           "ColumnSpan":3
        },
        "ControlLoc_Row":2,
        "ControlLoc_Column":1,
        "ControlLoc_ColumnSpan":3,
        "ControlUse":3,
        "ControlValue":1,
        "ControlString":"",
        "ControlStringList":null,
        "ControlStringSelected":null,
        "ControlFlag":false
     }
  ],
   "ref":2256,
   "name":"light",
   "location":"Office",
   "location2":"First Floor"
}
```

Where:

Label = The label to display on this control, if its a button, it w
the button label like "On" or "Off"
ControlType = Specifies the control type for this control, poss
values are:

```
 Not_Specified = 1
        Values = 2                         'This is the default to use if o
the others is not specified.
        Single_Text_from_List = 3
        List_Text_from_List = 4
        Button = 5
        ValuesRange = 6                  'Rendered as a drop-list by
        ValuesRangeSlider = 7
        TextList = 8
        TextBox_Number = 9
        TextBox_String = 10
        Radio_Option = 11
        Button_Script = 12      ' Rendered as a button, executes a
when activated.
        Color_Picker = 13
```

ControlLocation = Specifies the desired location of the contro
row/column
ControlUse = If the pair is to control a specifc function, such as
On/Off/Dim, this specifies this function. This makes it easier to
basic devices without knowing the label. For example, some de
may use "On" for On, and others may use "Bright Full" for on. By
checking this property it is easy to find the control element for c
controls. The possible values are:

```
    Not_Specified = 0
    _On = 1
    _Off = 2
    _Dim = 3
    _On_Alternate = 4
    _Play = 5                ' media control devices
    _Pause = 6
    _Stop = 7
    _Forward = 8
    _Rewind = 9
    _Repeat = 10
    _Shuffle = 11
```

ControlValue = The value for this pair, use when controlling the
device.

Range = Specifies the range of this pair. This is used for pairs
represent multiple values, such as the dim level of a light. A ligh
specifiy a range of 1-99 for the dim levels. The range also spec
the label using RangeStatusPrefix (such as "Dim"), and
RangeStatusSuffix (such as "%"). The formatted status of the dev
then be formatted as "Dim ## %".

/JSON?request=controldevicebyvalue&ref=###&value=#	Control a device given the device's reference number "ref", and "value". For example, if a light has a value of 0 for off, the follow would turn off the device with reference # 3570: /JSON?request=controldevicebyvalue&ref=3570&value=0 The return is the current JSON formatted status of the device (s return as "getstatus"), or the string "error".
/JSON?request=controldevicebylabel&ref=###&label=label	Control a device by label, this is the label as returned by the "getcontrol" parameter. For example, if the device has a label "(turn a device on, the following URL would turn it on: /JSON?request=controldevicebylabel&ref=3570&label=On The return is the current JSON formatted status of the device (s return as "getstatus"), or the string "error".

/JSON?request=getevents	Returns the names of all events in the system. An event is an ac be performed such as controlling a light, a sequence of lights, a thermostat, etc. Events have two properties, a group name and event name. This command returns the group name and event for all events. These two pieces of information are used to conti event.
	
	Eaxmple:
	
	`{`
	`   "Name":"HomeSeer Events",`
	`   "Version":"1.0",`
	`   "Events":[`
	`      {`
	`         "Group":"Lighting",`
	`         "Name":"Outside Lights Off"`
	`      }, (MORE EVENTS FOLLOW)`

/JSON? request=runevent&group=GROUPNAME&name=EVENTNAME OR /JSON?request=runevent&id=EVENT_ID	This command will execute the actions of an event. Pass the grc name and event name. The group and name are not case sensi event may also be run using the event ID.
/JSON?request=speak&phrase=text&host=HOST:NAME	Speaks the given phrase using text-to-speech. phrase = the phrase to speak host = the speaker host to speak out of. HomeSeer supports m hosts, like PC's and mobile devices. Each device is assgined a u host:name ID. For example, a host on the PC named "hometrolle the name "Android" would have the host name: HomeTroller:An this is added to the host parameter, then the phrase will be spc that host only. Many hosts can be added and are separated by a comma, IE: host=HomeTroller:Android,iPhone:bill
/JSON?request=getlocations	Returns all the location names for location 1 and location 2
/JSON?request=getcounter&counter=NAME	Returns the value for the given named counter

/JSON?request=getsetting&setting=SETTING_NAME	Returns the value for a specific settting. For example, the settir the name of location 1 is called "gLocLabel". To get the name of label use:
	
	/JSON?request=getsetting&setting=gLocLabel
	
	The return might be:
	
	`{`
	`"Value":"Room"`
	`}`

POST requests

JSON post requests can be used to control devices. The following request will turn a device on using the device value. In this example, a value of 255 will turn this device on:

Post data: {'action' : 'controlbyvalue', 'deviceref' : '3570', 'value' : '255'}
Url: ip_address/JSON

To control a device by label. In this case the label "On" will turn the device on:

Post data: {'action' : 'controlbylabel', 'deviceref' : '3570', 'label' : 'on'}

To run the actions of an event:

Post data: {'action' : 'runevent', 'group' : 'GROUPNAME', 'name' : 'EVENTNAME'}

See Also

Document Revisions
Getting Started
Plugin Initialization
Base Plugin API
Devices
Callbacks
Triggers
Actions
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater
Appendices

Home > Plugin Initialization

Plugin Initialization

Articles in this section

Plug-ins are EXE files, normally they are console applications that do not display any windows. HomeSeer launches the EXE without a console window but one can be shown if the developer mode is enabled on the Interfaces page. When HomeSeer starts it scans the HomeSeer folder for any EXE files that match a specific format. All plugins must be named as:

HSPI_PLUGINNAME.exe

If the name of your plugin is "AcmeWidget", then the EXE will named "HSPI_AcmeWidget.exe". Note the filename is case sensitive in that HomeSeer will use the filename to find the namespace in your plugin. So in this case the namespace in your .NET project would be set to "HSPI_AcmeWidget". You need a top level class named "HSPI" that will hold the main API that HomeSeer will access. At startup HomeSeer will use .NET reflection to load your EXE into memory and call a few key functions to get information about your plugin. This is done so your plug-in will appear on the HomeSeer interfaces page. See the following sections for more information.

See Also

Document Revisions
Getting Started
Controlling with JSON
Base Plugin API
Devices
Callbacks
Triggers
Actions
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater
Appendices

Home > Plugin Initialization > Finding Your Plugin

Finding Your Plugin

The following functions are mandatory and the plugin will not load if these do not exist. These functions are all part of your HSPI class. If you add:

```
Implements PlugInApi
```

in your HSPI class the VB.NET IDE will add all the required functions for you.

HSCOMPort() as Boolean

Your plug-in should return True if you wish to use HomeSeer's interfaces page of the configuration for the user to enter a serial port number for your plug-in to use. If enabled, HomeSeer will return this COM port number to your plug-in in the InitIO call. If you wish to have your own configuration UI for the serial port, or if your plug-in does not require a serial port, return False.

Name() as String

Probably one of the most important properties, the Name function in your plug-in is what the plug-in is identified with at all times. The filename of the plug-in is irrelevant other than when HomeSeer is searching for plug-in files, but the Name property is key to many things, including how plug-in created triggers and actions are stored by HomeSeer. If this property is changed from one version of the plug-in to the next, all triggers, actions, and devices created by the plug-in will have to be re-created by the user. Please try to keep the Name property value short, e.g. 14 to 16 characters or less. Web pages, trigger and action forms created by your plug-in can use a longer, more elaborate name if you so desire. In the sample plug-ins, the constant **IFACE_NAME** is commonly used in the program to return the name of the plug-in. No spaces or special characters are allowed other than a dash or underscore.

Capabilities() as Integer

The capabilities are what tell HomeSeer the plug-in type. Constants for these types are as follows:

```
CA_IO        = 4     (All plugins must have this set)
CA_THERM     = 16    (uses the thermostat API)
CA_MUSIC     = 32    (uses the music API)
```

Or these values together to create the value. All plugins must return at least 4. If you are using the thermostat API then return (4 or 16) or 20.

AccessLevel() as Integer

This determines whether the plug-in is free, or is a licensed plug-in using HomeSeer's licensing service. Return a value of 1 for a free plug-in, a value of 2 indicates that the plug-in is licensed using HomeSeer's licensing.

See Also

Initialization
Accessing Other Plug-Ins

Initialization

If your plug-in is configured to be used in the system, and if it is a licensed plug-in and a valid license to use the plug-in exists, then the plug-in will be fully initialized with the following series of HomeSeer--Plug-In exchanges.

InitIO(port As Integer) As String

If your plugin is set to start when HomeSeer starts, or is enabled from the interfaces page, then this function will be called to initialize your plugin. If you returned TRUE from HSComPort then the port number as configured in HomeSeer will be passed to this function. Here you should initialize your plugin fully. The hs object is available to you to call the HomeSeer scripting API as well as the callback object so you can call into the HomeSeer plugin API. HomeSeer's startup routine waits for this function to return a result, so it is important that you try to exit this procedure quickly. If your hardware has a long initialization process, you can check the configuration in InitIO and if everything is set up correctly, start a separate thread to initialize the hardware and exit InitIO. If you encounter an error, you can always use InterfaceStatus to indicate this.

ShutdownIO()

When HomeSeer shuts down or a plug-in is disabled from the interfaces page this function is then called. You should terminate any threads that you started, close any COM ports or TCP connections and release memory. After you return from this function the plugin EXE will terminate and must be allowed to terminate cleanly.

See Also

Finding Your Plugin
Accessing Other Plug-Ins

Accessing Other Plug-Ins

There may be times when access to another plug-in is required. For example, a plugin that creates a user interface may need to access a thermostat or music plugin. Each plugin has a direct TCP connection to HomeSeer and cannot connect to other plugins. A wrapper class is provided that allows HomeSeer to transfer information between plugins. To acces another plugin create an instance of the folliowing class:

```
Dim pa As New PluginAccess(hs, "Z-Wave", "")
```

This creates a new PluginAccess object and attempts to connect to the given plugin. If successful, the Connected property will be set to True. Going forward, you can access the plugin using this object.

The following function will request access to the Z-Wave plugin and access its name property and interface status:

```
    Private Sub AccessPlugin()
        Dim pa As New PluginAccess(hs, "Z-Wave", "")
        If pa.Connected Then
            hs.WriteLog(IFACE_NAME, "Connected to plugin Z-Wave")
            hs.WriteLog(IFACE_NAME, "Interface name: " & pa.Name & " Interface status: " &
pa.InterfaceStatus.intStatus.ToString)
        Else
            hs.WriteLog(IFACE_NAME, "Could not connect to plugin Z-Wave, is it running?")
        End If
    End Sub
```

See Also

Finding Your Plugin
Initialization

Home > Base Plugin API

Base Plugin API

Articles in this section

A plugin implements one more interfaces or API's in order to support specific functions. All plugins must implement the IPluginAPI. The following additional API''s are available:

IThermostatAPI	(for thermostats)
IMusicAPI	(for music systems)

These API's are implemented in the HSPI class as:

```
Public Class HSPI
    Implements IPluginAPI

End Class
```

A reference to the HomeSeerAPI file is needed also:

Imports HomeSeerAPI

Plugin API functions:

Information/Initialization API

ReadOnly Property Name As String
Function Capabilities() As Integer
ReadOnly Property HSCOMPort() As Boolean
Function AccessLevel() As Integer
Function SupportsMultipleInstances() As Boolean
Function InstanceFriendlyName() As String
Function InterfaceStatus() As Integer
Sub HSEvent(ByVal EventType As Enums.HSEvent, ByVal parms() As Object)
Sub ShutdownIO()
Function RaisesGenericCallbacks() As Boolean
Sub SetIOEx(ByVal dv As Object, ByVal housecode As String, ByVal devicecode As String, ByVal command As Integer, ByVal brightness As Integer, ByVal data1 As Integer, ByVal data2 As Integer, ByVal voice_command As String, ByVal host As String)
Function InitIO(ByVal port As Integer) As String
Function PollDevice(ByVal dvref As Integer, ByVal Address As String) As Double
Function SupportsConfigDevice() As Boolean
Function SupportsConfigDeviceAll() As Boolean
Function ConfigDevicePost(ByVal ref As Integer, ByVal data As String, ByVal user As String, ByVal userRights As Integer) As Boolean
Function ConfigDevice(ByVal ref As Integer, ByVal user As String, ByVal userRights As Integer) As String
Sub ButtonPress(ByVal button_name As String, ByVal dv As Object)

HS3 Actions/Triggers/Conditions

Function ActionCount() As Integer
Property ActionAdvancedMode() As Boolean
ReadOnly Property ActionName(ByVal ActionNumber As Integer) As String
Function ActionConfigured(ByVal ActInfo As strTrigActInfo) As Boolean
Function ActionBuildUI(ByVal sUnique As String, ByVal ActInfo As strTrigActInfo) As String
Function ActionProcessPostUI(ByVal PostData As Collections.Specialized.NameValueCollection, ByVal TrigInfoIN As strTrigActInfo) As strMultiReturn
Function ActionFormatUI(ByVal ActInfo As strTrigActInfo) As String
Function ActionReferencesDevice(ByVal ActInfo As strTrigActInfo, ByVal dvRef As Integer) As Boolean
Function HandleAction(ByVal ActInfo As IPlugInAPI.strTrigActInfo) As Boolean

Function Search(ByVal SearchString As String, ByVal RegEx As Boolean) As Boolean

ReadOnly Property HasConditions(ByVal TriggerNumber As Integer) As Boolean
ReadOnly Property HasTriggers() As Boolean

ReadOnly Property TriggerCount() As Integer
ReadOnly Property SubTriggerCount(ByVal TriggerNumber As Integer) As Integer

ReadOnly Property TriggerName(ByVal TriggerNumber As Integer) As String
ReadOnly Property SubTriggerName(ByVal TriggerNumber As Integer, ByVal SubTriggerNumber As Integer) As String

ReadOnly Property TriggerConfigured(ByVal TrigInfo As strTrigActInfo) As Boolean
Function TriggerBuildUI(ByVal sUnique As String, ByVal TrigInfo As strTrigActInfo) As String
Function TriggerProcessPostUI(ByVal PostData As Collections.Specialized.NameValueCollection, ByVal TrigInfoIN As strTrigActInfo) As strMultiReturn
Function TriggerFormatUI(ByVal TrigInfo As strTrigActInfo) As String
Function TriggerTrue(ByVal TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo) As Boolean
Function TriggerReferencesDevice(ByVal TrigInfo As strTrigActInfo, ByVal dvRef As Integer) As Boolean

Property Condition(ByVal TrigInfo As strTrigActInfo) As Boolean

Web Page

Function GenPage(ByVal link As String) As String
Function PagePut(ByVal data As String) As String
Function GetPagePlugin(ByVal page As String, ByVal user As String, ByVal userRights As Integer, ByVal queryString As String) As String
Function PostBackProc(ByVal page As String, ByVal data As String, ByVal user As String, ByVal userRights As Integer) As String

User Defined

Function PluginProc(ByVal procName As String, ByVal parms() As Object) As Object

See Also

Document Revisions
Getting Started
Controlling with JSON
Plugin Initialization
Devices
Callbacks
Triggers
Actions
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater
Appendices

Home > Base Plugin API > Name

Name

Public ReadOnly Property Name As String Implements HomeSeerAPI.IPlugInAPI.Name

Returns the name of your plug-in. This is used to identify your plug-in to HomeSeer and your users. Keep the name to 16 characters or less. Do not access any hardware in this function as HomeSeer will call this function using .NET reflection when it scans all plug-in EXE files so it should only return the text string of your plug-in.

Do NOT use special characters in your plug-in name with the exception of "-", ".", and " " (space).

Parameters: None
Returns: String

See Also

Capabilities
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Search
Licensing

Capabilities

`Public Function Capabilities() As Integer Implements HomeSeerAPI.IPlugInAPI.Capabilities`

Return the API's that this plug-in supports. This is a bit field. All plug-ins must have bit 3 set for I/O. This value is 4. Other possbile API values are:

```
Enum eCapabilities As Integer
        CA_IO = 4
        CA_Security = 8
        CA_Thermostat = &H10
        CA_Music = &H20
        CA_SourceSwitch = &H40
End Enum
```

Parameters: None

Returns: Integer capability

Example:

To return a capability which indicates the plug-in handles Security and includes support for at least one Thermostat, the return would be:

```
Public Function Capabilities() As Integer Implements HomeSeerAPI.IPlugInAPI.Capabilities
    Return HomeSeerAPI.Enums.eCapabilities.CA_IO Or Enums.eCapabilities.CA_Security Or Enums.eCapabilities
End Function
```

See Also

Name
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Search
Licensing

AccessLevel

`Public Function AccessLevel() As Integer Implements HomeSeerAPI.IPlugInAPI.AccessLevel`

Return the access level of this plug-in. Access level is the licensing mode. The following modes are available:

1 = Plug-in is not licensed and may be enabled and run without purchasing a license. Use this value for free plug-ins.
2 = Plug-in is licensed and a user must purchase a license in order to use this plug-in. When the plug-in is first enabled, it will will run as a trial for 30 days.

Parameters: None

Returns: Integer access level

See Also

Name
Capabilities
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Search
Licensing

InstanceFriendlyName

`Public Function InstanceFriendlyName() As String Implements HomeSeerAPI.IPlugInAPI.InstanceFriendlyName`

Returns the instance name of this instance of the plug-in. Only valid if SupportsMultipleInstances returns TRUE. The instance is set when the plug-in is started, it is passed as a command line parameter. The initial instance name is set when a new instance is created on the HomeSeer interfaces page. A plug-in needs to associate this instance name with any local status that it is keeping for this instance. See the multiple instances section for more information.

Parameters: None

Returns: A string that is the instance name

See Also

Name
Capabilities
AccessLevel
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Search
Licensing

SupportsMultipleInstances

`Public Function SupportsMultipleInstances() As Boolean Implements`

`HomeSeerAPI.IPlugInAPI.SupportsMultipleInstances`

Return TRUE if the plug-in supports multiple instances. The plug-in may be launched multiple times and will be passed a unique instance name as a command line parameter to the Main function. The plug-in then needs to associate all local status with this particular instance.

This feature is ideal for cases where multiple hardware modules need to be supported. For example, an single irrigation controller supports 8 zones but the user needs 16. They can add a second controller as a new instance to control 8 more zones. This assumes that the second controller would use a different COM port or IP address.

When multiple instances are enabled, HomeSeer will allow the user to add another instance on the Interfaces page. Each instance will be displayed as a seperate line.

The instance is passed to the main function in the plugin and should be saved for future reference.

To register different web pages for each instance, change the link that is created with hs.RegisterLink and hs.RegisterConfigLink like so:

```
   Dim wpd As New webPageDesc
    wpd.link = IFACE_NAME & instance                ' we add the instance so it goes to the proper plugin
instance when selected

    If instance <> "" Then
        wpd.linktext = "Sample Plugin Config instance " & instance
    Else
        wpd.linktext = "Sample Plugin Config"
    End If
    wpd.page_title = "Sample Plugin Config"
    wpd.plugInName = IFACE_NAME
    wpd.plugInInstance = instance
    callback.RegisterConfigLink(wpd)
```

Parameters: None

Returns: TRUE if supporting multiple instances.

See Also

- Name
- Capabilities
- AccessLevel
- InstanceFriendlyName
- HSComPort
- SetIOMulti
- InterfaceStatus
- InitIO
- ShutdownIO
- Custom Functions (Script Commands)
- Search
- Licensing

Home > Base Plugin API > HSComPort

HSComPort

`Public ReadOnly Property HSCOMPort As Boolean Implements HomeSeerAPI.IPlugInAPI.HSCOMPort`

Return True if HomeSeer is to manage the communications (COM) port for this plug-in. A COM port selection box will be displayed on the interfaces page so the user can enter a COM port number. If the plug-in supports multiple instances, the COM port will be managed for each instance. When InitIO is called in the plug-in the selected COM port will passed as a parameter.

Parameters: None

Returns: True or False

See Also

Name
Capabilities
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Search
Licensing

Home > Base Plugin API > SetIOMulti

SetIOMulti

`Public Sub SetIOMulti(colSend As System.Collections.Generic.List(Of HomeSeerAPI.CAPI.CAPIControl)) Implements HomeSeerAPI.IPlugInAPI.SetIOMulti`

SetIOMulti is called by HomeSeer when a device that your plug-in owns is controlled. Your plug-in owns a device when it's INTERFACE property is set to the name of your plug-

The parameters passed to SetIOMulti are as follows - depending upon what generated the SetIO call, not all parameters will contain data. Be sure to test for "Is Nothing" before testing for values or your plug-in may generate an exception error when a variable passed is uninitialized.

colsend- This is a collection of CAPIControl objects, one object for each device that needs to be controlled. Look at the ControlValue property to get the value that device needs to be set to.

Example to extract the value from each CAPIControl object in the colSend collection. This example also shows how to notify HomeSeer of the change. If setting the value in the hardware fails, do not call hs.SetDeviceValueByRef and log a warning or error message.

```
Dim CC As CAPIControl
For Each CC In colSend
     Console.WriteLine("SetIOMulti set value: " & CC.ControlValue.ToString & "->ref:" & CC.Ref.ToString)
     hs.SetDeviceValueByRef(CC.Ref, CC.ControlValue, True)
   Next
```

See Also

Name
Capabilities
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Search
Licensing

Home > Base Plugin API > InterfaceStatus

InterfaceStatus

`Public Function InterfaceStatus() As Integer Implements HomeSeerAPI.IPlugInAPI.InterfaceStatus`

HomeSeer may call this function at any time to get the status of the plug-in. Normally it is displayed on the Interfaces page. The return is an object that represents the status. The object is of type HomeSeerAPI.IPlugInAPI.strInterfaceStatus

Parameters: None
Returns: HomeSeerAPI.IPlugInAPI.strInterfaceStatus

Example:

```
Public Function InterfaceStatus() As HomeSeerAPI.IPlugInAPI.strInterfaceStatus Implements
HomeSeerAPI.IPlugInAPI.InterfaceStatus
        Dim es As New IPlugInAPI.strInterfaceStatus
        es.intStatus = IPlugInAPI.enumInterfaceStatus.OK
        Return es
End Function
```

See Also

- Name
- Capabilities
- AccessLevel
- InstanceFriendlyName
- SupportsMultipleInstances
- HSComPort
- SetIOMulti
- InitIO
- ShutdownIO
- Custom Functions (Script Commands)
- Search
- Licensing

InitIO

`Public Function InitIO(ByVal port As Integer) As String Implements HomeSeerAPI.IPlugInAPI.InitIO`

Called to initialize your plug-in. Initialize your hardware/software, start any threads and return an error status.

Parameters: port number
The port number is the COM port number your hardware is using. This port number is only valid if your plug-in returns TRUE for HSCOMPort property.

Returns: An empty string if the plug-in initializes ok, else an error message that will be displayed in the HomeSeer event log. If you return an error string, your plug-in will be unloaded and will not be running. If your initialization detects an error that you can recover from later, then use hs.writelog to log a message to the HomeSeer log and return an empty string. This will keep your plug-in loaded and running.

See Also

- Name
- Capabilities
- AccessLevel
- InstanceFriendlyName
- SupportsMultipleInstances
- HSComPort
- SetIOMulti
- InterfaceStatus
- ShutdownIO
- Custom Functions (Script Commands)
- Search
- Licensing

ShutdownIO

`Public Sub ShutdownIO() Implements HomeSeerAPI.IPlugInAPI.ShutdownIO`

Called when HomeSeer is not longer using the plug-in. This call will be made if a user disables a plug-in from the interfaces configuration page and when HomeSeer is shut down.

Parameters: None

Returns: Nothing

See Also

Name
Capabilities
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
Custom Functions (Script Commands)
Search
Licensing

Custom Functions (Script Commands)

There may be times when you need to offer a custom function that is not part of the plugin API. The following API functions allow users to call your plugin from scripts and web pages by calling the functions by name. The functions are:

```
Public Function PluginFunction(ByVal proc As String, ByVal parms() As Object) As Object Implements
IPlugInAPI.PluginFunction
Public Function PluginPropertyGet(ByVal proc As String, parms() As Object) As Object Implements
IPlugInAPI.PluginPropertyGet
Public Sub PluginPropertySet(ByVal proc As String, value As Object) Implements IPlugInAPI.PluginPropertySet
```

You do not have to modify these functions. Simply copy them from the sample plugin.

These functions allow scripts to call any function in your plugin by name. For example, if you added a function name "GetVoltagePoint(point as integer)", a script can call your function with:

```
' get a reference to the plugin
plugin = New HomeSeerAPI.PluginAccess(hs, "VoltagePlugin", "")

' call the function and get the value
dim voltage as integer = plugin.PluginFunction("GetVoltagePoint",{1})
```

See Also

Name
Capabilities
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Search
Licensing

Search

```
Public Function Search(ByVal SearchString As String,
                                   ByVal RegEx As Boolean) As HomeSeerAPI.SearchReturn() Implements
IPlugInAPI.Search
```

This procedure will be called in your plug-in by HomeSeer whenever the user uses the search function of HomeSeer, and your plug-in is loaded and initialized. Unlike ActionReferencesDevice and TriggerReferencesDevice, this search is not being specific to a device, it is meant to find a match anywhere in the resources managed by your plug-in. This could include any textual field or object name that is utilized by the plug-in.

Parameters:

SearchString: (String) The string to search for.

RegEx: (Boolean) True if the search string is a regular expression.

Returns: Array of SearchReturn items describing what was found and where it was found. In general, return values are:

- A URL with the properly formed URL being provided in the RValue structure member.
- An object, with the object being in the RObject structure member.
- Any other return provided as a string in the RValue structure member.

The RType structure member should indicate which type of return each item is (see eSearchReturn Enum)
The RDescription structure member should be used to provide a description of the return item and/or where it was found.

Example:

```
RET = New SearchReturn
RET.RType = eSearchReturn.r_String_Other
RET.RDescription = "Found in the zone description for zone 4"
RET.RValue = Zone(4)
colRET.Add(RET)

Return colRET.ToArray
```

References:

```
<Serializable()> _
Public Enum eSearchReturn
    r_String_Other = 0
    r_URL = 1
    r_Object = 2
End Enum

<Serializable()> _
Public Structure SearchReturn
    Dim RType As eSearchReturn
    Dim RDescription As String
    Dim RValue As String
    Dim RObject As Object
End Structure
```

See Also

Name
Capabilities
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Licensing

Licensing

If your plug-in is a free plug-in then all you need to do is return 1 in your AccessLevel function. This instructs HomeSeer that the plug-in does not require a license and anyone will be able to enable it.

If your plug-in is a paid plug-in then return 2 from AccessLevel. HomeSeer will then allow the plug-in to run as a trial for 30 days. The trial starts the first time the user enables the plug-in. After 30 days, the user will need to enter a license ID and password, obtained from the HomeSeer store, in order to continue to use the plug-in.

See Also

Name
Capabilities
AccessLevel
InstanceFriendlyName
SupportsMultipleInstances
HSComPort
SetIOMulti
InterfaceStatus
InitIO
ShutdownIO
Custom Functions (Script Commands)
Search

Home > Devices

Devices

Articles in this section

The information regarding properties of devices and the script commands that affect them are all documented in the HomeSeer help system. However, what you don't know is how devices, and their various settings, affect their operation with plug-ins, and how those relate to HomeSeer. This section describes the features of devices and their interaction with HomeSeer and plug-ins beyond what the help file shows.

A device displays it's status through the use of status/values/graphics pairs. For example, a device that can be ON or OFF will have 2 pairs associated with it, one pair has a status text of "On" and an associated value of "1", and the second has a text status of "Off" and an associated value of "0". When this device is displayed on the HomeSeer status page it will display with 2 buttons, one labeled "On" and the other "Off". When a user clicks the "On" button, the value property of the device will go to a 1. The status/value pairs for any device can be viewed and altered by clicking on the "Edit Status Graphics" from the device properties page (Device Utitliy screen).

Devices created by plug-ins use value/status pairs, as status value pairs determine what options appear in the control drop-down list on the device status web page. If the status of a status/value pair is preceded with a tilde (~), it indicates to HomeSeer that it is a valid status to display if the device value is at the appropriate value, but it indicates that the status is NOT a control option. Thus, with a tilde in front of Dim: On = 1, Off = 0, ~Dim = 2. With these set on the device, I can set the value to 1 and it will display a status of "On", but over on the right hand side of the device status web page, the drop down list of control options will only list On and Off as choices.

Functionality similar to that of status/value pairs exists for associating graphical images with values, and those are referred to as graphics/value pairs.

One of the properties of a device (See the HomeSeer help file, Device Class, for the complete device class property list) is the ***interface*** property. When you set a device's interface property to the name of your plug-in, you are taking ownership for that device. When a user changes a device using the HomeSeer UI or a script command and it is owned by a plug-in, the owning plug-in's SetIO function is called. The plug-in can then examine the information provided with the SetIO call and invoke a change on the hardware or software interface it is controlling. The plug-in may then choose to update the device again itself to reflect that the command was carried out successfully.

See Also

Document Revisions
Getting Started
Controlling with JSON
Plugin Initialization
Base Plugin API
Callbacks
Triggers
Actions
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater
Appendices

Home > Devices > DeviceClass Related

DeviceClass Related

This section of the Devices documentation refers specifically to properties or objects which are a part of the DeviceClass object - in other words, they are a part of the devices themselves.

See Also

PollDevice
Device Control API (CAPI)
Configuring Devices from UI

Creating Devices

To create a device in your plug-in, you can use the same hs commands that you use in scripts. However, to create an I/O device which is typically owned by a plug-in, then there are some special commands you would use to accomplish this.

When you create your device(s), keep the following in mind:

- Devices have a CODE field and an ADDRESS field. The CODE field used to be the old housecode/unitcode field in HS2 abd is just a string value. Address is whatever you put there, and in the HomeSeer UI, it automatically appends the CODE field. If you call a procedure that accepts a device string as a parameter, it will only match on the CODE field. In the HomeSeer application API there are functions to get a device reference by address or device reference number. Use of these fields are optional as all devices have a unique reference number assigned to them and this reference number can always be used to find and access a device. The CODE is useful for accessing devices through the use of scripts. To get a CODE value call hs.GetNextVirtualCode.
- Use the IOMISC string property of a device to hold information about the device specific to your plug-in. If the device is user created, created by another plug-in, or a shared device, do not use IOMISC as you may overwrite another application's information. If the device is to be owned by your plug-in, you can use IOMISC to put information about the device right in the device itself, thus (as an example of a good use) the user can completely change the name, location, or location2 properties and you can still find the device.
    - At startup, get a DeviceEnumerator object and process all of the devices in the system.
    - Examine the device, and if the interface property is set to the name of your plug-in, then examine the IOMISC property to determine the type of device it is for your plug-in.
    - When you find a device, store its device reference ID in an array or some other internal means for use in modifying the device while your plug-in is running.
- Set the INTERFACE property to the name of your plug-in. This is how HomeSeer knows that your plug-in owns that device. When the device is controlled, and depending upon other settings, the SetIO procedure in your plug-in will be called.
- Set the MISC property of the device accordingly. If you are modifying an existing device to work with your plug-in, remember that MISC is an option bit property, and so modifications should be done using AND and OR logic operations.
- If you do not wish to use device value/status pairs and only want some control buttons, set the buttons property of the device.
- After you receive a CODE and ADDRESS for your devices, you may want save this code in your INI file. When HomeSeer is restarted, you can retrieve this housecode and therefore know where your devices are located. See the sample plug-in for how this may be done. The only issue with this method is that its possible for a user to delete your INI file which will force you to re-build your devices, which may cause duplicates. Always search for your devices first to verify that they exist. Enumerate all the devices and see if any devices have the INTERFACE property set to your plugin name. An alternate method is to simply create your devices initially then search for them when HomeSeer starts. When you find the first one, you now have the housecode for all your devices. There is no need to save the housecode in this case, and you do not need to be concerned about your INI file being deleted.
- If you create devices that you intend to manage with device value/status pairs exclusively, and are not going to update their status using hs.SetDeviceStatus commands, then you may wish to add the MISC_NO_STATUS_TRIG property to the MISC property.
  (dv.misc = dv.misc OR MISC_NO_STATUS_TRIG)
  This will prevent that device from appearing in the drop down list of devices on the device status trigger screen, as well as prevent the device from appearing in the list of devices for the device conditions, which depend upon device status.
  (There are other device options set by the MISC property values - See the MISC values in HomeSeer Constants .)

See the sample plugin for an example on how to create devices.

See Also

The Device Class
PlugExtraData
AdditionalDisplayData
ScaleText / HasScale

The Device Class

Please see the "Devices" section of the Scripting documentation for information regarding the Device Class, Device Value/Status Pairs, Device Value/Graphic Pairs, and Device_Type.

See Also

Creating Devices
PlugExtraData
AdditionalDisplayData
ScaleText / HasScale

PlugExtraData

The device class contains an additional object class which can be used by plug-ins to store "Extra Data". This object store consists of named or unnamed objects. Accessing these objects is done through the top-level object clsPlugExtraData, which may be SET or retrieved with a GET from the device class object through the read-only property PlugExtraData_Get and the write-only property PlugExtraData_Set.

As with other device class properties being accessed through the communications interface, the object passes by value one-way through the interface, so changes need to be done by providing the application interface object (typically called 'hs') so that the change may be done internally by HomeSeer.

Example:

```
 Accessing the Extra Data object to work with it...
  Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
  EDO = dv.PlugExtraData_Get(hs)

 Setting the modified Extra Data object back to the device...
  dv.PlugExtraData_Set(hs) = EDO
```

** After calling ".._Set", call hs.SaveEventsDevices to force HomeSeer to save the device that was modified.

See Also

Creating Devices
The Device Class
AdditionalDisplayData
ScaleText / HasScale

NamedCount

This procedure in the clsPlugExtraData object returns the count of the number of named Extra Data Objects in the EDO storage.

```
Public Function NamedCount() As Integer
```

Example:

```
  Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
  EDO = dv.PlugExtraData_Get(hs)
  Dim Count As Integer
  Count = EDO.NamedCount
```

See Also

UnNamedCount
AddNamed
AddUnNamed
RemoveNamed
RemoveUnNamed
GetNamed
GetNamedKey
GetNamedKeys
GetUnNamed
GetAllUnNamed
ClearAllNamed
ClearAllUnNamed

UnNamedCount

This procedure in the clsPlugExtraData object returns the count of the number of unnamed Extra Data Objects in the EDO storage.

`Public Function UnNamedCount() As Integer`

Example:

```
Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
EDO = dv.PlugExtraData_Get(hs)
Dim Count As Integer
Count = EDO.UnNamedCount
```

See Also

NamedCount
AddNamed
AddUnNamed
RemoveNamed
RemoveUnNamed
GetNamed
GetNamedKey
GetNamedKeys
GetUnNamed
GetAllUnNamed
ClearAllNamed
ClearAllUnNamed

AddNamed

This procedure in the clsPlugExtraData object adds a named object to the EDO storage.

`Public Function AddNamed(ByVal Key As String, ByVal Obj As Object) As Boolean`

Example:

```
Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
EDO = dv.PlugExtraData_Get(hs)
If Not EDO.AddNamed("My Special Object", MyObject) Then
            ' An error occurred.
End If
' Now put the modified EDO back.
dv.PlugExtraData_Set(hs) = EDO
            hs.SaveEventsDevices()   ' << Device was changed, so tell HomeSeer to save it now.
```

Note: If the key entry already exists, this will NOT overwrite it! Make sure you remove a named entry before you attempt to update it.

See Also

NamedCount
UnNamedCount
AddUnNamed
RemoveNamed
RemoveUnNamed
GetNamed
GetNamedKey
GetNamedKeys
GetUnNamed
GetAllUnNamed
ClearAllNamed
ClearAllUnNamed

AddUnNamed

This procedure in the clsPlugExtraData object adds an unnamed object to the EDO storage.

Public Function AddUnNamed(ByVal Obj As Object) As Boolean

Example:

```
Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
EDO = dv.PlugExtraData_Get(hs)
Dim Index As Integer
Index = EDO.AddUnNamed(MyObject)
' Now put the modified EDO back.
dv.PlugExtraData_Set(hs) = EDO
        hs.SaveEventsDevices()   ' << Device was changed, so tell HomeSeer to save it now.
```

See Also

NamedCount
UnNamedCount
AddNamed
RemoveNamed
RemoveUnNamed
GetNamed
GetNamedKey
GetNamedKeys
GetUnNamed
GetAllUnNamed
ClearAllNamed
ClearAllUnNamed

RemoveNamed

This procedure in the clsPlugExtraData object, removes a named object from the EDO storage.

Public Function RemoveNamed(ByVal Key As String) As Boolean

Example:

```
Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
EDO = dv.PlugExtraData_Get(hs)
If Not EDO.RemoveNamed("My Special Object") Then
        ' An error occurred.
End If
' Now put the modified EDO back.
dv.PlugExtraData_Set(hs) = EDO
        hs.SaveEventsDevices()   ' << Device was changed, so tell HomeSeer to save it now.
```

See Also

- NamedCount
- UnNamedCount
- AddNamed
- AddUnNamed
- RemoveUnNamed
- GetNamed
- GetNamedKey
- GetNamedKeys
- GetUnNamed
- GetAllUnNamed
- ClearAllNamed
- ClearAllUnNamed

RemoveUnNamed

This procedure in the clsPlugExtraData object removes an unnamed object from the EDO storage. This procedure is overloaded:
   Public Function RemoveUnNamed(ByVal Index As Integer) As Boolean
   Public Function RemoveUnNamed(ByVal Obj As Object) As Boolean

```
Public Function RemoveUnNamed(ByVal Index As Integer) As Boolean
Public Function RemoveUnNamed(ByVal Obj As Object) As Boolean
```

Example:

```
 Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
 EDO = dv.PlugExtraData_Get(hs)
           Dim Index As Integer = 29
 If Not EDO.RemoveUnNamed(Index) Then
           ' An error occurred, try the other method?
  If Not EDO.RemoveUnNamed(MyObject) Then
   ' Just not my lucky day...
  End If
 End If
 ' Now put the modified EDO back.
 dv.PlugExtraData_Set(hs) = EDO
           hs.SaveEventsDevices()   ' << Device was changed, so tell HomeSeer to save it now.
```

Note: If you call RemoveUnNamed using an object, and the object is a numerical data type (especially Integer), it will be interpreted as the Index.

See Also

- NamedCount
- UnNamedCount
- AddNamed
- AddUnNamed
- RemoveNamed
- GetNamed
- GetNamedKey
- GetNamedKeys
- GetUnNamed
- GetAllUnNamed
- ClearAllNamed
- ClearAllUnNamed

GetNamed

This procedure in the clsPlugExtraData object retrieves a named object from the EDO storage.

`Public Function GetNamed(ByVal Key As String) As Object`

Example:

```
 Dim EDO As DeviceAPI.clsPlugExtraData = Nothing
        EDO = dv.PlugExtraData_Get(hs)
        If EDO IsNot Nothing Then
        Dim obj As Object = Nothing
               obj = EDO.GetNamed("My Special Object")
               If obj IsNot Nothing Then
               Log("Plug-In Extra Data Object Retrieved = " & obj.ToString, LogType.LOG_TYPE_INFO)
               End If
```

See Also

- NamedCount
- UnNamedCount
- AddNamed
- AddUnNamed
- RemoveNamed
- RemoveUnNamed
- GetNamedKey
- GetNamedKeys
- GetUnNamed
- GetAllUnNamed
- ClearAllNamed
- ClearAllUnNamed

GetNamedKey

This procedure in the clsPlugExtraData object retrieves they key for a named object from the EDO storage using its index.

`Public Function GetNamedKey(ByVal Index As Integer) As String`

Example:

```
 Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
 EDO = hs.PlugExtraData_Get(hs)
 If EDO IsNot Nothing Then
  If EDO.NamedCount > 0 Then
   For i As Integer = 0 to EDO.NamedCount - 1
    Log "They key at Index " & i.ToString & " is:" & EDO.GetNamedKey(i)
   Next
  End If
 End If
```

See Also

- NamedCount
- UnNamedCount
- AddNamed
- AddUnNamed
- RemoveNamed
- RemoveUnNamed
- GetNamed
- GetNamedKeys
- GetUnNamed
- GetAllUnNamed
- ClearAllNamed
- ClearAllUnNamed

GetNamedKeys

This procedure in the clsPlugExtraData object retrieves all keys for named objects in the EDO storage.

```
Public Function GetNamedKeys() As String()
```

Example:

```
 Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
 EDO = hs.PlugExtraData_Get(hs)
 If EDO IsNot Nothing Then
  If EDO.NamedCount > 0 Then
   Dim Keys() As String = Nothing
   Keys = EDO.GetNamedKeys
   If Keys IsNot Nothing AndAlso Keys.Count > 0 Then
    For i As Integer = 0 to Keys.Length - 1
     Log "One of the named keys is:" & Keys(i)
    Next
   End If
  End If
 End If
```

See Also

- NamedCount
- UnNamedCount
- AddNamed
- AddUnNamed
- RemoveNamed
- RemoveUnNamed
- GetNamed
- GetNamedKey
- GetUnNamed
- GetAllUnNamed
- ClearAllNamed
- ClearAllUnNamed

GetUnNamed

This procedure in the clsPlugExtraData object retrieves an unnamed object from the EDO storage.

```
Public Function GetUnNamed(ByVal Index As Integer) As Object
```

Example:

```
  Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
  EDO = dv.PlugExtraData_Get(hs)
               Dim Index As Integer = 29
  Dim obj As Object = Nothing
  obj = EDO.GetUnNamed(Index)
  If obj IsNot Nothing Then
   ' I found my object!
  End If
```

See Also

- NamedCount
- UnNamedCount
- AddNamed
- AddUnNamed
- RemoveNamed
- RemoveUnNamed
- GetNamed
- GetNamedKey
- GetNamedKeys
- GetAllUnNamed
- ClearAllNamed
- ClearAllUnNamed

GetAllUnNamed

This procedure in the clsPlugExtraData object retrieves all unnamed objects in the EDO storage.

Public Function GetAllUnNamed() As Object()

Example:
```
 Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
 EDO = hs.PlugExtraData_Get(hs)
 If EDO IsNot Nothing Then
  If EDO.UnNamedCount > 0 Then
   Dim objs() As Object = Nothing
   objs = EDO.GetAllUnNamed
   If objs IsNot Nothing AndAlso objs.Count > 0 Then
    ' Do whatever you want with the objects...
   End If
  End If
 End If
```

See Also

NamedCount
UnNamedCount
AddNamed
AddUnNamed
RemoveNamed
RemoveUnNamed
GetNamed
GetNamedKey
GetNamedKeys
GetUnNamed
ClearAllNamed
ClearAllUnNamed

ClearAllNamed

This procedure in the clsPlugExtraData object clears (deletes) all named objects in the EDO storage.

Public Function ClearAllNamed(ByVal Confirm As Boolean) As Boolean

Example:
```
 Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
 EDO = hs.PlugExtraData_Get(hs)
 If EDO IsNot Nothing Then
  If EDO.NamedCount > 0 Then
   EDO.ClearAllNamed(True)
   hs.PlugExtraData_Set(hs) = EDO
   hs.SaveEventsDevices
  End If
 End If
```

See Also

NamedCount
UnNamedCount
AddNamed
AddUnNamed
RemoveNamed
RemoveUnNamed
GetNamed
GetNamedKey
GetNamedKeys
GetUnNamed
GetAllUnNamed
ClearAllUnNamed

ClearAllUnNamed

This procedure in the clsPlugExtraData object clears (deletes) all unnamed objects in the EDO storage.

```
Public Function ClearAllUnNamed(ByVal Confirm As Boolean) As Boolean
```

Example:
```
 Dim EDO as DeviceAPI.clsPlugExtraData = Nothing
 EDO = hs.PlugExtraData_Get(hs)
 If EDO IsNot Nothing Then
  If EDO.UnNamedCount > 0 Then
   EDO.ClearAllUnNamed(True)
   hs.PlugExtraData_Set(hs) = EDO
   hs.SaveEventsDevices
  End If
 End If
```

See Also

NamedCount
UnNamedCount
AddNamed
AddUnNamed
RemoveNamed
RemoveUnNamed
GetNamed
GetNamedKey
GetNamedKeys
GetUnNamed
GetAllUnNamed
ClearAllNamed

AdditionalDisplayData

The AdditionalDisplayData (See the scripting reference for the prototype.) allows you to set several replacement variables upon a device's value/status pairs so that information not available at design time when the pairs are created can still be displayed.

Consider this scenario... A Z-Wave Smoke Detector can report that it detects smoke, and it can include a location name so that you know where the smoke was detected without having to look at cross reference maps of installation locations.   Since the location name is variable and is provided only at alarm time, or may not be included at all, you can provide the location in the AdditionalDisplayData prior to setting the device to the alarm value, and the data will be substituted where indicated in the pair.

Example:

Set up a pair for the value 1 to have a status of "Fire Alarm".
Set up another pair for the value 2 to have a status of "Fire Alarm issued @%0@"
Set up a third pair for the value 3 to have a status of "Fire Alarm issued @%0@ at location @%1@"

When the alarm happens, and if the exact time of the alarm cannot be determined, set the device to a value of 1.
When the alarm happens and you do know the time, do this instead:

```
Dim s(0) As String
s(0) = Now.ToString
dv.AdditionalDisplayData(hs) = s
hs.SetDeviceValueByRef(dv.Ref(hs), 2, True)
```

The replacement variable at index 0, indicated in the status pair by @%0@, will be replaced with the first element in the AdditionalDisplayData array, which was set to the current time.

When the alarm happens and you know the time and also get location information:

```
Dim s(1) As String
```

```
s(0) = Now.ToString
s(1) = Alarm.Location
dv.AdditionalDisplayData(hs) = s
hs.SetDeviceValueByRef(dv.Ref(hs), 3, True)
```

The replacement variable at index 0, indicated in the status pair by @%0@, will be replaced with the first element in the AdditionalDisplayData array, which was set to the current time, and the variable at index 1 (@%1@) will be replaced with the second element which is the location information.

The Value/Status pairs contain a function which will generate the replacement variable string for you. After creating a new VSPair object, reference it and the AddDataReplace function to generate a replacement variable. The prototype of the function is:

```
Public Shared Function AddDataReplace(ByVal Index As Integer) As String
```

Example:

To set up the pair that uses the date and location in the above example:

```
Pair = New VSPair(HomeSeerAPI.ePairStatusControl.Status)
Pair.PairType = VSVGPairType.SingleValue
Pair.Value = 3
Pair.IncludeValues = False
Pair.Status = "Fire Alarm issued " & VSPair.AddDataReplace(0) & _
                  " at location " & SPair.AddDataReplace(1)
Default_VS_Pairs_AddUpdateUtil(dv.Ref(hs), Pair)
```

This results in the status being: Fire Alarm issued @%0@ at location @%1@
It is not required to use AddDataReplace, it is provided simply as a utility function.

See Also

Creating Devices
The Device Class
PlugExtraData
ScaleText / HasScale

Home > Devices > DeviceClass Related > ScaleText / HasScale

ScaleText / HasScale

The ScaleText (See the scripting reference for the prototype.) allows you to set a scale or other indicator upon a device's value/status pairs so that information not available at design time when the pairs are created can still be displayed.

Consider this scenario... A Z-Wave temperature sensor on a thermostat displays the temperature in Fahrenheit, but then the user changes the thermostat to use Celsius scale instead. One way to address this is to remove and re-add the value/status pair for the temperature to change F to C, or another way - especially if you are unaware of when the user might change the scale - is to use the ScaleText replacement in the value/status pair.

The ScaleText is always used to replace any occurrence of the string @S@ in the device status.

To use this feature, HasScale in the value/status pair must also be set to True.

Example:

Set up a pair for the range of values a temperature can be and to use the ScaleText feature...

```
Pair = New VSPair(HomeSeerAPI.ePairStatusControl.Status)
Pair.PairType = VSVGPairType.Range
Pair.RangeStart = -2147483648
Pair.RangeEnd = 2147483647
Pair.IncludeValues = True
Pair.ValueOffset = 0
Pair.RangeStatusDecimals = 1
Pair.RangeStatusPrefix = ""
Pair.RangeStatusSuffix = " " & VSPair.ScaleReplace     <------
Pair.HasScale = True                                   <------
```

```
Default_VS_Pairs_AddUpdateUtil(dv.Ref(hs), Pair)
```

When the temperature changes and you know the scale, update the device as follows:

```
' Set the ScaleText to our scale so that @S@ in the value/status pair will
'   get replaced with our scale read from the temperature reading.
dv.ScaleText(hs) = TemperatureReading.TempScale
hs.SetDeviceValueByRef(dv.Ref(hs), TemperatureReading.Temperature, True)
```

The Value/Status pairs contain a constant which will provide you with the replacement variable string for you. After creating a new VSPair object, reference it and in the status use:

```
Public Const ScaleReplace As String = "@S@"
```

Using ScaleReplace is not required, the replacement string may be typed out as well.

To indicate to HomeSeer that a ScaleText exists, remember to also set HasScale to True in the value/status pair.

See Also

Creating Devices
The Device Class
PlugExtraData
AdditionalDisplayData

PollDevice

`Public Function PollDevice(ByVal dvref As Integer) As Double Implements HomeSeerAPI.IPlugInAPI.PollDevice`

If a device is owned by your plug-in (interface property set to the name of the plug-in) and the device's status_support property is set to True, then this procedure will be called in your plug-in when the device's status is being polled, such as when the user clicks "Poll Devices" on the device status page.

Normally your plugin will automatically keep the status of its devices updated. There may be situations where automatically updating devices is not possible or CPU intensive. In these cases the plug-in may not keep the devices updated. HomeSeer may then call this function to force an update of a specific device. This request is normally done when a user displays the status page, or a script runs and needs to be sure it has the latest status.

Parameters:

dvRef: device reference #

Returns: new value of the device

See Also

DeviceClass Related
Device Control API (CAPI)
Configuring Devices from UI

Device Control API (CAPI)

Please see the "Devices" section of the Scripting documentation for information on the Device Control API (CAPI).

See Also

DeviceClass Related
PollDevice
Configuring Devices from UI

Configuring Devices from UI

This section describes plug-in functions that are used to configure devices from the device utility page in HomeSeer.

See Also

DeviceClass Related
PollDevice
Device Control API (CAPI)

SupportsConfigDevice

```
Public Function SupportsConfigDevice() As Boolean Implements HomeSeerAPI.IPlugInAPI.SupportsConfigDevice
```

Return TRUE if your plug-in allows for configuration of your devices via the device utility page. This will allow you to generate some HTML controls that will be displayed to the user for modifying the device.

Parameters: None

Returns: TRUE or FALSE

See Also

SupportsConfigDeviceAll
ConfigDevice
ConfigDevicePost
SupportsAddDevice

SupportsConfigDeviceAll

```
Public Function SupportsConfigDeviceAll() As Boolean Implements HomeSeerAPI.IPlugInAPI.SupportsConfigDeviceAll
```

If your plug-in manages all devices in the system, you can return TRUE from this function. Your configuration page will be available for all devices.

Parameters: None

Returns: TRUE or FALSE

See Also

SupportsConfigDevice
ConfigDevice
ConfigDevicePost
SupportsAddDevice

ConfigDevice

`ConfigDevice(ref As Integer, user As String, userRights As Integer, newDevice as Boolean) As String Implements HomeSeerAPI.IPlugInAPI.ConfigDevice`

If SupportsConfigDevice returns TRUE, this function will be called when the device properties are displayed for your device. The device properties is displayed from the Device Utility page. This page displays a tab for each plug-in that controls the device. Normally, only one plug-in will be associated with a single device. If there is any configuration that needs to be set on the device, you can return any HTML that you would like displayed. Normally this would be any jquery controls that allow customization of the device. The returned HTML is just an HTML fragment and not a complete page.

If the newDevice parameter is TRUE, the user is adding a new device from the HomeSeer user interface. If you return TRUE from your SupportsAddDevice then ConfigDevice will be called when a user is creating a new device. Your tab will appear and you can supply controls for the user to create a new device for your plugin. When your ConfigDevicePost is called you will need to get a reference to the device using the past ref number and then take ownership of the device by setting the interface property of the device to the name of your plugin. You can also set any other properties on the device as needed.

Parameters:

ref: The device reference #
user: The user that is logged into the server and viewing the page
userRights: The rights of the logged in user
newDevice: True if this a new device being created for the first time. In this case, the device configuration dialog may present different information than when simply editing an existing device.

Returns: A string containing HTML to be displayed. Return an empty string if there is not configuration needed.

See Also

SupportsConfigDevice
SupportsConfigDeviceAll
ConfigDevicePost
SupportsAddDevice

ConfigDevicePost

`Public Function ConfigDevicePost(ref As Integer, data As String, user As String, userRights As Integer) As Enums.ConfigDevicePostReturnImplements HomeSeerAPI.IPlugInAPI.ConfigDevicePost`

This function is called when a user posts information from your plugin tab on the device utility page. The function has the following possible return values:

```
    DoneAndSave = 1             Any changes to the config are saved and the page is closed and the user it
returned to the device utility page
    DoneAndCancel = 2           Changes are not saved and the user is returned to the device utility page
    DoneAndCancelAndStay = 3    No action is taken, the user remains on the plugin tab
    CallbackOnce = 4            Your plugin ConfigDevice is called so your tab can be refereshed, the user
stays on the plugin tab
    CallbackTimer = 5           Your plugin ConfigDevice is called and a page timer is called so
ConfigDevicePost is called back every 2 seconds
```

Here is a sample postback procedure that detects when a button with an id of "mybutton" is pressed:

```
    Public Function ConfigDevicePost(ref As Integer, data As String, user As String, userRights As Integer) As
Enums.ConfigDevicePostReturn Implements HomeSeerAPI.IPlugInAPI.ConfigDevicePost


        Dim parts As Collections.Specialized.NameValueCollection
        parts = HttpUtility.ParseQueryString(data)
        If parts("id") = "mybutton" Then
        ' do something
        End If
        Return Enums.ConfigDevicePostReturn.CallbackOnce
    End Function
```

Parameters:
ref: Device reference number
data: query string data posted to the web server (name/value pairs from controls on the page)
user: logged in user
userRights: rights of logged in user

Returns: True if no error, else False

See Also

SupportsConfigDevice
SupportsConfigDeviceAll
ConfigDevice
SupportsAddDevice

Home > Devices > Configuring Devices from UI > SupportsAddDevice

SupportsAddDevice

`Public Function SupportsAddDevice() As Boolean Implements HomeSeerAPI.IPlugInAPI.SupportsAddDevice`

Return TRUE if the plugin supports the ability to add devices through the Add Device link on the device utility page. If TRUE a tab appears on the add device page that allows the user to configure specific options for the new device.

When ConfigDevice is called the newDevice parameter will be True if this is the first time the device config screen is being displayed and a new device is being created. See ConfigDevicePost for more information.

See Also

SupportsConfigDevice
SupportsConfigDeviceAll
ConfigDevice
ConfigDevicePost

Callbacks

Articles in this section

See Also

GetNextVirtualAddress

Public Function GetNextVirtualAddress() As String Implements IHSApplication.GetNextVirtualCode

This will return the next free virtual address that you an assign to your device. This address can be used by scripts to reference your device. It is not required to assign a virtual address to your devices as all devices are assigned a unique reference number that can be used by your application to find and access your devices.

Usage of the callback is as follows:

```
strCode = hs.GetNextVirtualAddress
```

HomeSeer supports up to 25974 devices via device codes A1 through Z with 999 codes available for each letter code.

See Also

RegisterEventCB

Public Sub RegisterEventCB(ByVal evtype As Enums.HSEvent, ByVal PIName As String, ByVal PIInstance As String) Implements IAppCallbackAPI.RegisterEventCB

Call this function when your plugin initializes to notify HomeSeer that you want to be called when a specific event happens. The normal use for this is to be notified when a device changes value or it's displayed string changes. You will be notified about any device change, not just changes to your own devices. However, if your device is controlled and your SetIOMulti() call is made, after you call back with hs.SetDeviceValueByRef you will get an HSEvent notifying you about the value change. Since the change is to your device, this notification should be ignored.

When an event is detected that has beenn registered by your plug-in, call is made to the HSEvent function in your plug-in. You can then handle the event. See HSEvent for more information and an example.

The following events can be requested and are defined in the HSApplicationAPI file in the enum named HSEvent.

Event Callback Constants and Purpose

Event Name	Value (Hex)	Value (Decimal)	Description
not used	1	1	
EV_TYPE_LOG	2	2	A message written to the event log
not used	4	4	
AUDIO	8	8	Audio start or stop
not used	10	16	
CONFIG_CHANGE	20	32	Device or event has changed
STRING_CHANGE	40	64	Device's string value has changed
SPEAKER_CONNECT	80	128	Speaker client connected to HomeSeer
CALLER_ID	100	256	Caller ID Information
not used	200	512	
VALUE_CHANGE	400	1024	Device's value has changed
VOICE_REC	400	1024	Future addition, not yet implemented

GENERIC	8000	32768	Generic event raised by other plug-ins and scripts.

```
Enum HSEvent
    LOG = 2
    AUDIO = 8
    CONFIG_CHANGE = &H20
    STRING_CHANGE = &H40
    SPEAKER_CONNECT = &H80
    CALLER_ID = &H100
    VALUE_CHANGE = &H400
    VOICE_REC = &H1000
    Unused_2000 = &H2000
    Unused_4000 = &H4000
    GENERIC = &H8000
End Enum
```

For example, in InitIO the following call can be made to register for any change to a device's value:

```
callback.RegisterEventCB(Enums.HSEvent.VALUE_CHANGE, IFACE_NAME, "")
```

See Also

GetNextVirtualAddress
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

HSEvent

`Public Sub HSEvent(ByVal EventType As Enums.HSEvent, ByVal parms() As Object) Implements HomeSeerAPI.IPlugInAPI.HSEvent`

When you wish to have HomeSeer call back in to your plug-in or application when certain events happen in the system, call the RegisterEventCB procedure and provide it with event you wish to monitor. See RegisterEventCB for more information and an example and event types.

The parameters are passed in an array of objects. 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 in the table in RegisterEventCB.

Parameter Events

Parameters for the Event Log event are:

parms(1)=date_time	Date and time of log entry as a string.
parms(2)=msg_class	Type of log entry, i.e., "speak" or "ERROR".

parms(3)=msg	The actual event log message.
parm(4)=color	Color of log entry.
parm(5)=Pri	Priority of entry.
parm(6)=From	String describing where the log came from.
parm(7)=ErrorCode	A number which is the error code, for error entries.
parm(8)=Date	Date object that is the date of the log entry. Used for localization.

Parameters for the Audio event are:

parms(1)=started	TRUE means that the audio channel is now being used (HomeSeer is speaking or a WAV file is playing). When True, it indicates when there is audio activity at any of the speaker apps. For instance, if an event is to go to all speaker apps and there are 3 apps running, then 3 will be returned. If an event goes to a specific speaker app then only 1 will be returned. FALSE means that the device is now free.
parms(2)=typeid	This is the type of audio that is being played. 0 is for TTS (HomeSeer speaking) and 100 is for WAV files.
parms(3)=host	speaker host name
parms(4)=instance	speaker host instance

Parameters for the Configuration Change event are:

parms(1)=type	0=a device was changed 1=an event was changed 2=an event group was changed.
parms(2)=id	The index ID of the device or event that was change. This has been deprecated and may return any value.
parms(3)=ref	The device reference number, event reference number, or event group reference number. 0 means that the reference is not known.
parms(4)=DAC	Whether the device/event was deleted, added, or changed. 0=not known, 1=Added, 2=Deleted, 3=Changed
parms(5)=WhatChanged	A string describing what changed.

To get a reference to the actual device that was modified, use set dv = hs.GetDeviceByRef(ref).
To get a reference to the actual event that was modified, use set ev = hs.GetEventByRef(ref).

The ID may be 0, which means either multiple events or devices were modified, the device or event was deleted, or the device or event was modified from a script, which means the device/event ID was not known.

Parameters for the Device String Change event are:

parms(1)=address	The address of the device that changed (string value).
parms(2)=txt	The new value for the device string (what it's changing to).
parms(3)=dvref	Device reference number (integer)

Parameters for the Speaker Client Connect event are:

parms(1)=host	The host computer (name) that the speaker client is on.
parms(2)=instance	The instance name of the speaker client connecting - if an instance name is not specified, the speaker client uses 'Default' as the instance name.
parms(3)=cStatus	Connection Status is 0 for disconnect, 1 for connect.
parms(4)=IPAddr	The IP address of the host machine where the speaker client resides.

Speaker client disconnects may not always register due to network problems.

Parameters for the Caller ID event are:

parms(1)=CIDNumber	The Caller ID reported phone number from the phone company.
parms(2)=CIDName	The Caller ID reported caller name from the phone company.
parms(3)=dir_last	Last name from the entry in the address book matching CIDNumber.
parms(4)=dir_first	First name from the entry in the address book matching CIDNumber.
parms(5)=dir_company	Company name from the entry in the address book matching CIDNumber.
parms(6)=line	The line number that the CID information was obtained from.

You must have the Caller ID Name service in order to receive CIDName data, otherwise this will be blank.

Parameters for the device Value change event are:

parms(1)=address	The device's address (string)

parms(2)=new value	The new value of the device (double).
parms(3)=old value	The old value of the device (double).
parms(4)=dvRef	The device's reference number (integer)

Parameters for the Generic Type event are:

parms(1)=Generic Event	The generic event string describing this event type. Note: An asterisk may be used to designate any/all generic events in plug-ins.
parms(2)=Sender	The plug-in name (if an object was used) or name string (if called from a script) that identifies the plug-in or script that raised this event.
parms(3...n)=Parms	The parameters that were provided by the plug-in or script that called RaiseGenericEventCB.

See Also

GetNextVirtualAddress
RegisterEventCB
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

RaiseGenericEventCB

```
Public Sub RaiseGenericEventCB(ByVal GenericType As String, ByVal Parms() As Object, ByVal PIName As String,
ByVal PIInstance As String) Implements IAppCallbackAPI.RaiseGenericEventCB
```

Purpose

When an application or plug-in registers to receive specific types of generic HSEvent callbacks, this procedure is used to raise those callbacks and send information to that application. See RegisterGenericEventCB , UnRegisterGenericEventCB ,and HSEvent information for more details.

Parameters

Parameter: **Generic Type**
Type: **string**
Description: This is the generic type name that was used when the receiving plug-in or application called RegisterGenericEventCB. If you

know that the plug-in or application that you wish to raise the generic event with used an asterisk as the Generic Type, then you can use any text here as that plug-in will receive all generic event callbacks.

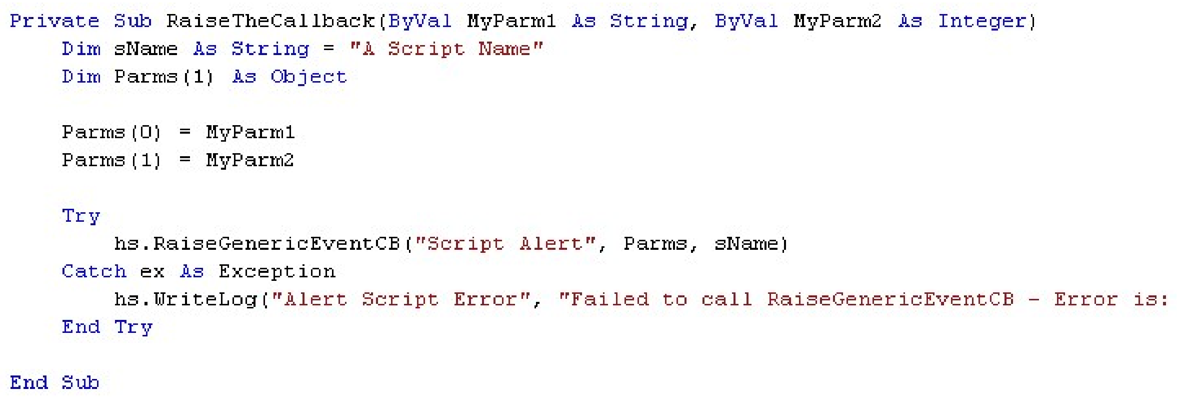
Parameter: **Parms()**
Type: **Array of Objects**
Description: These are parameters that you wish to be passed to the receiving application. As an array of objects, it can contain strings, integers, other objects, etc.

Parameter: **Object ID**
Type: **object or string**
Description: This provides the name of the plug-in or application raising the event. If an object is provided, then the object's Name() procedure will be called to get the object name. If a string is provided, then that will be passed to the receiving application verbatim. When used with the scripting interface, always provide a string object.

Returns

None.

Example



```
Private Sub RaiseTheCallback(ByVal MyParm1 As String, ByVal MyParm2 As Integer)
    Dim sName As String = "A Script Name"
    Dim Parms(1) As Object

    Parms(0) = MyParm1
    Parms(1) = MyParm2

    Try
        hs.RaiseGenericEventCB("Script Alert", Parms, sName)
    Catch ex As Exception
        hs.WriteLog("Alert Script Error", "Failed to call RaiseGenericEventCB - Error is:
    End Try

End Sub
```

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

Home > Callbacks > RegisterGenericEventCB

RegisterGenericEventCB

Public Sub RegisterGenericEventCB(ByVal GenericType As String, ByVal PIName As String, ByVal PIInstance As String) Implements IAppCallbackAPI.RegisterGenericEventCB

Purpose

HomeSeer has the ability to raise events in applications and plug-ins when one of a list of specific events in HomeSeer occurs (See RegisterEventCB). RegisterGenericEventCB allows an application or plug-in writer the opportunity to have custom events raised and to enable other applications and plug-ins to receive those callbacks.

To remove the callback script, call UnRegisterGenericEventCB.

Parameters

Parameter: **Generic Type**
Type: **string**

Description: This is a string that identifies the callback. For example, a type of "MyPlugEvent" would mean that calls to RaiseGenericEventCB using something other than "MyPlugEvent" would be ignored. This string should be unique, and should be provided to all applications wishing to register to receive these callbacks. A special value of a single asterisk (*) can be used to indicate that you wish to receive ALL generic type callbacks from other plug-ins/applications.

Parameter: **PIName**
Type: String

Description: This is the name of the plugin that will accept the event.

Parameter: **PIInstance**
Type: String

Descripting: This is the plugin instance that will accept the event, enter and empty string if the target plugin does not support mulitiple instances.

Returns

None.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

Home > Callbacks > RegisterProxySpeakPlug

RegisterProxySpeakPlug

Public Sub RegisterProxySpeakPlug(ByVal PIName As String, ByVal PIInstance As String) Implements IAppCallbackAPI.RegisterProxySpeakPlug

This callback registers your plug-in as a Speak Proxy plug-in. The PIName parameter is the name of your plug-in, normally IFACE_NAME, and the PIInstance is the plugin instance name.

After this registration, whenever a Speak command is issued in HomeSeer, your plug-in's SpeakIn procedure will be called instead. When your plug-in wishes to have HomeSeer actually speak something, it uses SpeakProxy instead of Speak.

If you no longer wish to proxy Speak commands in your plug-in, or when your plug-in has its Shutdown procedure called, use callback.UnRegisterProxySpeakPlug to remove the registration as a speak proxy.

See Also

- GetNextVirtualAddress
- RegisterEventCB
- HSEvent
- RaiseGenericEventCB
- RegisterGenericEventCB
- ReplaceVariables
- UnRegisterGenericEventCB
- UnRegisterProxySpeakPlug
- RaisesGenericCallbacks
- TriggerFire
- GetTriggers
- GetTriggersInst
- TriggerMatches
- TriggerMatchesInst
- Structures Used

Home > Callbacks > ReplaceVariables

ReplaceVariables

Public Function ReplaceVariables(ByVal sIn As String) As String

HomeSeer supports the use of replacement variables, which is the use of special tags to indicate where HomeSeer should replace the tag with text information. A full list of replacement variables is listed
in HomeSeer's help file. Plug-ins can use this feature directly by calling ReplaceVariables via the plug-in callback interface.

Pass into this call a string with the replacement variables, and the returned string will have the replacement variables removed with the indicated values put in their place.

Example:

```
    Dim stNew as string = callback.ReplaceVariables("The time is $$time")
```

$$time will be replaced with the current time.

See Also

- GetNextVirtualAddress
- RegisterEventCB
- HSEvent
- RaiseGenericEventCB
- RegisterGenericEventCB
- RegisterProxySpeakPlug
- UnRegisterGenericEventCB
- UnRegisterProxySpeakPlug
- RaisesGenericCallbacks
- TriggerFire
- GetTriggers
- GetTriggersInst
- TriggerMatches
- TriggerMatchesInst
- Structures Used

Home > Callbacks > UnRegisterGenericEventCB

UnRegisterGenericEventCB

```
Public Sub UnRegisterGenericEventCB(ByVal GenericType As String, ByVal PIName As String, ByVal PIInstance As
String) Implements IAppCallbackAPI.UnRegisterGenericEventCB
```

Purpose

This will remove an application or plug-in from the list that should receive generic event callbacks for the type indicated (See RegisterGenericEventCB).

Parameters

Parameter: **Generic Type**
Type: **string**
Description: This is the generic type string that was used to register the callback with RegisterGenericEventCB.

Parameter: **PIName**
Type: **string**
Description: Name of the plugin.

Parameter: **PIInstance**
Type: **string**
Description: Name of the plugin instance.

Returns

None.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

Home > Callbacks > UnRegisterProxySpeakPlug

UnRegisterProxySpeakPlug

```
Public Sub UnRegisterProxySpeakPlug(ByVal PIName As String, ByVal PIInstance As String) Implements
IAppCallbackAPI.UnRegisterProxySpeakPlug
```

This callback is used to unregister a plug-in as a Speak proxy that was previously registered using RegisterProxySpeakPlug. The name parameter is the plug-in name, which is usually IFACE_NAME.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

RaisesGenericCallbacks

Function RaisesGenericCallbacks() As Boolean

HSTouch uses the Generic Event callback in some music plug-ins so that it can be notified of when a song changes, rather than having to repeatedly query the music plug-in for the current song status. If this property is present (and returns True), especially in a Music plug-in, then HSTouch (and other plug-ins) will know that your HSEvent procedure can handle generic callbacks.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

TriggerFire

Sub TriggerFire(ByVal Plug_Name As String, ByVal TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo)

This function is a callback function and is called when a plugin detects that a trigger condition is true. The passed parameters are:

Plug_Name: The interface name of the plugin that is triggering

TrigInfo: The TrigInfo structure of the trigger that is triggering

Example from the Sample Basic plugin when a weight value is greater than the specified weight in the trigger:

```
If Weight > Trig1.TriggerWeight Then
```

```
 Log("Weight trigger is TRUE - calling FIRE! for event ID " & TC.evRef.ToString, LogType.LOG_TYPE_WARNING)
 callback.TriggerFire(IFACE_NAME, TC)
  ' Step 3: If a trigger matches, call FIRE!
End If
```

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

Home > Callbacks > GetTriggers

GetTriggers

```
Function GetTriggers(ByVal PIName As String) As HomeSeerAPI.IPlugInAPI.strTrigActInfo()
```

Callback function to get a list of triggers for a plugin. If the plug-in has multiple instances, the Instance in the strTrigActInfo structure will contain the name of the instance.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggersInst
TriggerMatches
TriggerMatchesInst
Structures Used

Home > Callbacks > GetTriggersInst

GetTriggersInst

```
Function GetTriggersInst(ByVal Plug_Name As String, ByVal Plug_Inst As String) As
HomeSeerAPI.IPlugInAPI.strTrigActInfo()
```

Callback function to get a list of triggers for a plugin, but only the ones matching the plug-in instance name provided.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
TriggerMatches
TriggerMatchesInst
Structures Used

TriggerMatches

```
Function TriggerMatches(Plug_Name As String, TrigID As Integer, SubTrig As Integer) As
HomeSeerAPI.IPlugInAPI.strTrigActInfo()
```

This function returns an array of strTrigActInfo which matches the given plug-in, trigger number, and sub-trigger number provided. GetTriggers returns all triggers, so use TriggerMatches when you only want to know if there are triggers in events for a specific plug-in's trigger.

If the trigger does not have sub-triggers, use -1 for the SubTrig value.

If the plug-in supports multiple instances, the Instance property from the strTrigActInfo will contain the instance name for each returned value.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatchesInst
Structures Used

TriggerMatchesInst

```
Function TriggerMatchesInst(ByVal Plug_Name As String, ByVal Plug_Inst As String, ByVal TrigID As Integer,
ByVal SubTrig As Integer)
         As HomeSeerAPI.IPlugInAPI.strTrigActInfo()
```

```
This operates the same as TriggerMatches, with the addition of the Plug_Inst parameter to filter the returned
values to those matching the given instance name.
```

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
Structures Used

Structures Used

These two structures are used in Trigger and Action related callbacks and functions.

See Also

GetNextVirtualAddress
RegisterEventCB
HSEvent
RaiseGenericEventCB
RegisterGenericEventCB
RegisterProxySpeakPlug
ReplaceVariables
UnRegisterGenericEventCB
UnRegisterProxySpeakPlug
RaisesGenericCallbacks
TriggerFire
GetTriggers
GetTriggersInst
TriggerMatches
TriggerMatchesInst

strTrigActInfo

This is the prototype for the strTrigActInfo structure, which is used in many of the Trigger and Action related calls and callbacks from HomeSeer.

```
Structure strTrigActInfo

    ''' This is the unique event reference ID number for the event that this trigger is a part of.
    Public evRef As Integer

    ''' This is the unique ID for this trigger or action. When the trigger is true, the plug-in will pass this
    ''' to HomeSeer to trigger to cause HomeSeer to check the conditions and trigger the event if appropriate.
    ''' When the action needs to be carried out, HomeSeer will invoke the Handle procedure in the action, and
    ''' if there is action information stored by the plug-in, this property can be used to retrieve it.
    Public UID As Integer


    ''' This is for plug-ins only since plug-ins can support multiple types of different triggers or actions.
    ''' This identifies which, out of the triggers or actions offered by the plug-in, that this trigger
    ''' or action is.
    Public TANumber As Integer
    ''' When a trigger or action has sub-types, this is used to identify which sub-trigger or trigger sub-
type,
    ''' sub-action or action sub-type this trigger or action is set to. For example, in HomeSeer there is a
    ''' TIME trigger - the sub-ID might identify whether it is the type AT, BEFORE, or AFTER a time value.
    Public SubTANumber As Integer


    ''' This is exclusively for plug-ins. Using a serialization procedure, the data for this plug-in
    ''' can be stored and managed within the HomeSeer database by the plug-in storing the serialized
    ''' object here. When HomeSeer is calling into the plug-in and wants to provide the trigger object
    ''' for the plug-in to analyze, it provides the serialized object data here in this byte array which
    ''' the plug-in can use to de-serialize back into an object that holds the trigger info. Since
    ''' parameters cannot be passed through the interface ByRef, this is read-only, but the plug-in is
    ''' allowed to RETURN a structure that includes Data or DataOut, which is a byte array, and contains
    ''' the serialized object after the plug-in updated it or made changes to it.
    Public DataIn() As Byte

    Public Instance As String

End Structure
```

See Also

strMultiReturn

Home > Callbacks > Structures Used > strMultiReturn

strMultiReturn

This is a structure, which is used in the Trigger and Action ProcessPostUI procedures, which not only communications trigger and action information through TrigActInfo which is strTrigActInfo , but provides an array of Byte where an updated/serialized trigger or action object from your plug-in can be stored. See TriggerProcessPostUI and ActionProcessPostUI for more details.

```
Structure strMultiReturn

    ''' When plug-in calls such as ...BuildUI, ...ProcessPostUI, or ...FormatUI are called and there is
    ''' feedback or an error condition that needs to be reported back to the user, this string field
    ''' can contain the message to be displayed to the user in HomeSeer UI. This field is cleared by
    ''' HomeSeer after it is displayed to the user.
    Public sResult As String


    ''' This is the trigger or action info from HomeSeer - see the structure for more information.
    Public TrigActInfo As strTrigActInfo


    ''' (Also see DataIn of strTrigInfo) The serialization data for the plug-in object cannot be
    ''' passed ByRef which means it can be passed only one-way through the interface to HomeSeer.
    ''' If the plug-in receives DataIn, de-serializes it into an object, and then makes a change
    ''' to the object, this is where the object can be serialized again and passed back to HomeSeer
    ''' for storage in the HomeSeer database.
    Public DataOut() As Byte

End Structure
```

See Also

strTrigActInfo

Home > Triggers

Triggers

Articles in this section

HasConditions
HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
GetTriggers
TriggerMatches

See Also

Document Revisions
Getting Started
Controlling with JSON
Plugin Initialization
Base Plugin API
Devices
Callbacks
Actions
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater
Appendices

Home > Triggers > HasConditions

HasConditions

```
Public ReadOnly Property HasConditions(ByVal TriggerNumber As Integer) As Boolean Implements
HomeSeerAPI.IPlugInAPI.HasConditions
```

Return True if the given trigger can also be used as a condition, for the given trigger number.

Parameters:

TriggerNimber: (Integer)

Returns: Not supported, read-only property

See Also

HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
GetTriggers
TriggerMatches

Home > Triggers > HasTriggers

HasTriggers

```
Public ReadOnly Property HasTriggers() As Boolean Implements HomeSeerAPI.IPlugInAPI.HasTriggers
```

Return True if your plugin contains any triggers, else return false.

Parameters: None

Returns: Not supported, read-only property.

See Also

HasConditions
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
GetTriggers
TriggerMatches

Home > Triggers > TriggerTrue

TriggerTrue

```
Public Function TriggerTrue(ByVal TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo) As Boolean
```

Although this appears as a function that would be called to determine if a trigger is true or not, it is not. Triggers notify HomeSeer of trigger states using TriggerFire , but Triggers can also be conditions, and that is where this is used. If this function is called, TrigInfo will contain the trigger information pertaining to a trigger used as a condition. When a user's event is triggered and it has conditions, the conditions need to be evaluated immediately, so there is not regularity with which this function may be called in your plug-in. It may be called as often as once per second or as infrequently as once in a blue moon.

See Also

- HasConditions
- HasTriggers
- TriggerCount
- SubTriggerCount
- TriggerName
- SubTriggerName
- TriggerConfigured
- TriggerBuildUI
- TriggerProcessPostUI
- TriggerFormatUI
- Condition
- TriggerReferencesDevice
- TriggerFire
- GetTriggers
- TriggerMatches

Home > Triggers > TriggerCount

TriggerCount

`Public ReadOnly Property TriggerCount As Integer Implements HomeSeerAPI.IPlugInAPI.TriggerCount`

Return the number of triggers that your plugin supports.

Parameters: None

Returns: (Integer) Number of triggers.

See Also

- HasConditions
- HasTriggers
- TriggerTrue
- SubTriggerCount
- TriggerName
- SubTriggerName
- TriggerConfigured
- TriggerBuildUI
- TriggerProcessPostUI
- TriggerFormatUI
- Condition
- TriggerReferencesDevice
- TriggerFire
- GetTriggers
- TriggerMatches

Home > Triggers > SubTriggerCount

SubTriggerCount

`Public ReadOnly Property SubTriggerCount(ByVal TriggerNumber As Integer) As Integer Implements HomeSeerAPI.IPlugInAPI.SubTriggerCount`

Return the number of sub triggers your plugin supports.

Parameters: (Integer) TriggerNumber

Returns: Not supported, read-only property

See Also

HasConditions
HasTriggers
TriggerTrue
TriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
GetTriggers
TriggerMatches

TriggerName

`Public ReadOnly Property TriggerName(ByVal TriggerNumber As Integer) As String Implements HomeSeerAPI.IPlugInAPI.TriggerName`

Return the name of the given trigger based on the trigger number passed.

Parameters:

TriggerNumber: (Integer)

Returns: Not supported, read-only property.

See Also

HasConditions
HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
GetTriggers
TriggerMatches

SubTriggerName

`Public ReadOnly Property SubTriggerName(ByVal TriggerNumber As Integer, ByVal SubTriggerNumber As Integer) As String Implements HomeSeerAPI.IPlugInAPI.SubTriggerName`

Return the text name of the sub trigger given its trugger number and sub trigger number.

Parameters:

TriggerNumber: (Integer)

SubTriggerNumber: (Integer)

Returns: Not supported, read-only property

See Also

- HasConditions
- HasTriggers
- TriggerTrue
- TriggerCount
- SubTriggerCount
- TriggerName
- TriggerConfigured
- TriggerBuildUI
- TriggerProcessPostUI
- TriggerFormatUI
- Condition
- TriggerReferencesDevice
- TriggerFire
- GetTriggers
- TriggerMatches

TriggerConfigured

Public ReadOnly Property TriggerConfigured(ByVal TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo) As Boolean Implements HomeSeerAPI.IPlugInAPI.TriggerConfigured

Given a strTrigActInfo object detect if this this trigger is configured properly, if so, return True, else False.

Parameters:

TrigInfo: (HomeSeerAPI.IPlugInAPI.strTrigActInfo class)

Returns: Not supported, read-only property.

See Also

- HasConditions
- HasTriggers
- TriggerTrue
- TriggerCount
- SubTriggerCount
- TriggerName
- SubTriggerName
- TriggerBuildUI
- TriggerProcessPostUI
- TriggerFormatUI
- Condition
- TriggerReferencesDevice
- TriggerFire
- GetTriggers
- TriggerMatches

TriggerBuildUI

Public Function TriggerBuildUI(ByVal sUnique As String, ByVal TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo) As String Implements HomeSeerAPI.IPlugInAPI.TriggerBuildUI

Return HTML controls for a given trigger.

Parameters:

sUnique: (String)

TrigInfo: (HomeSeerAPI.IPlugInAPI.strTrigActInfo)

Returns: (String)

Note: If the combination of Trigger Number, Sub-Trigger Number, and Condition point to an invalid combination, an empty string ("") should be returned.

> **For example:** Your trigger reports that trigger 2 has conditions (HasConditions(2) = True), and trigger 2 has 4 sub-triggers, but only sub-triggers 2-4 can be conditions. When TriggerBuildUI is called, use the TrigInfo to get the trigger, and if the trigger number is 2, the sub-trigger is 1, and Condition has been set to True by HomeSeer, return "" from this procedure.

See Also

HasConditions
HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
GetTriggers
TriggerMatches

TriggerProcessPostUI

```
Public Function TriggerProcessPostUI(PostData As System.Collections.Specialized.NameValueCollection, _
                                     TrigInfoIn As HomeSeerAPI.IPlugInAPI.strTrigActInfo ) As
HomeSeerAPI.IPlugInAPI.strMultiReturn Implements HomeSeerAPI.IPlugInAPI.TriggerProcessPostUI
```

Process a post from the events web page when a user modifies any of the controls related to a plugin trigger. After processing the user selctions, create and return a strMultiReturn object. This object is defined as follows:

This is a structure, which is used in the Trigger and Action ProcessPostUI procedures, which not only communications trigger and action information through TrigActInfo which is strTrigActInfo , but provides an array of Byte where an updated/serialized trigger or action object from your plug-in can be stored. See TriggerProcessPostUI and ActionProcessPostUI for more details.

```
Structure strMultiReturn

    ''' When plug-in calls such as ...BuildUI, ...ProcessPostUI, or ...FormatUI are called and there is
    ''' feedback or an error condition that needs to be reported back to the user, this string field
    ''' can contain the message to be displayed to the user in HomeSeer UI. This field is cleared by
    ''' HomeSeer after it is displayed to the user.
    Public sResult As String


    ''' This is the trigger or action info from HomeSeer - see the structure for more information.
    Public TrigActInfo As strTrigActInfo


    ''' (Also see DataIn of strTrigInfo) The serialization data for the plug-in object cannot be
    ''' passed ByRef which means it can be passed only one-way through the interface to HomeSeer.
    ''' If the plug-in receives DataIn, de-serializes it into an object, and then makes a change
    ''' to the object, this is where the object can be serialized again and passed back to HomeSeer
    ''' for storage in the HomeSeer database.
    Public DataOut() As Byte

End Structure
```

Parameters:

PostData: (NameValueCollection) A collection of name value pairs that contains the values from the web page controls.

TrigInfoIn: (HomeSeerAPI.IPlugInAPI.strTrigActInfo)

Returns: (HomeSeerAPI.IPlugInAPi.strMultiReturn)

See Also

- HasConditions
- HasTriggers
- TriggerTrue
- TriggerCount
- SubTriggerCount
- TriggerName
- SubTriggerName
- TriggerConfigured
- TriggerBuildUI
- TriggerFormatUI
- Condition
- TriggerReferencesDevice
- TriggerFire
- GetTriggers
- TriggerMatches

TriggerFormatUI

```
Public Function TriggerFormatUI(TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo ) As String _
                                Implements HomeSeerAPI.IPlugInAPI.TriggerFormatUI
```

After the trigger has been configured, this function is called in your plugin to display the configured trigger. Return text that describes the given trigger.

Parameters:

TrigInfo: (HomeSeerAPI.IPlugInAPI.strTrigActInfo)

Returns: (String)

See Also

- HasConditions
- HasTriggers
- TriggerTrue
- TriggerCount
- SubTriggerCount
- TriggerName
- SubTriggerName
- TriggerConfigured
- TriggerBuildUI
- TriggerProcessPostUI
- Condition
- TriggerReferencesDevice
- TriggerFire
- GetTriggers
- TriggerMatches

Condition

```
Public Property Condition(TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo ) As Boolean Implements
HomeSeerAPI.IPlugInAPI.Condition
```

HomeSeer will set this to TRUE if the trigger is being used as a CONDITION. Check this value in BuildUI and other procedures to change how the trigger is rendered if it is being used as a condition or a trigger.

Parameters:

TrigInfo: (HomeSeerAPI.IPlugInAPI.strTrigActInfo)

Returns: (True or False)

See Also

- HasConditions
- HasTriggers
- TriggerTrue
- TriggerCount
- SubTriggerCount
- TriggerName
- SubTriggerName
- TriggerConfigured
- TriggerBuildUI
- TriggerProcessPostUI
- TriggerFormatUI
- TriggerReferencesDevice
- TriggerFire
- GetTriggers
- TriggerMatches

TriggerReferencesDevice

```
Public Function TriggerReferencesDevice(TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo , _
                                        dvRef As Integer) As Boolean _
                                        Implements HomeSeerAPI.IPlugInAPI.TriggerReferencesDevice
```

Return True if the given device is referenced by the given trigger.

Parameters:

TrigInfo: (strTrigActInfo)

dvRef: (Integer) The device reference number to check.

Example:

```
Dim Trig As MyTrig = Nothing
Try
   Trig = GetTrigger(TrigInfo)
Catch ex As Exception
   Trig = Nothing
End Try
If Trig Is Nothing Then Return False
Return Trig.Devices.Contains(dvRef)
```

See Also

HasConditions
HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerFire
GetTriggers
TriggerMatches

Home > Triggers > TriggerFire

TriggerFire

Sub TriggerFire(ByVal Plug_Name As String, ByVal TrigInfo As HomeSeerAPI.IPlugInAPI.strTrigActInfo)

This function is a callback function and is called when a plugin detects that a trigger condition is true. The passed parameters are:

Plug_Name: The interface name of the plugin that is triggering

TrigInfo: The TrigInfo structure of the trigger that is triggering

Example from the Sample Basic plugin when a weight value is greater than the specified weight in the trigger:

```
If Weight > Trig1.TriggerWeight Then
 Log("Weight trigger is TRUE - calling FIRE! for event ID " & TC.evRef.ToString, LogType.LOG_TYPE_WARNING)
 callback.TriggerFire(IFACE_NAME, TC)
  ' Step 3: If a trigger matches, call FIRE!
End If
```

See Also

HasConditions
HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
GetTriggers
TriggerMatches

Home > Triggers > GetTriggers

GetTriggers

```
Function GetTriggers(ByVal PIName As String) As HomeSeerAPI.IPlugInAPI.strTrigActInfo()
```

Callback function to get a list of triggers for a plugin. If the plug-in has multiple instances, the Instance in the strTrigActInfo structure will contain the name of the instance.

See Also

HasConditions
HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
TriggerMatches

Home > Triggers > TriggerMatches

TriggerMatches

```
Function TriggerMatches(Plug_Name As String, TrigID As Integer, SubTrig As Integer) As
HomeSeerAPI.IPlugInAPI.strTrigActInfo()
```

This function returns an array of strTrigActInfo which matches the given plug-in, trigger number, and sub-trigger number provided. GetTriggers returns all triggers, so use TriggerMatches when you only want to know if there are triggers in events for a specific plug-in's trigger.

If the trigger does not have sub-triggers, use -1 for the SubTrig value.

If the plug-in supports multiple instances, the Instance property from the strTrigActInfo will contain the instance name for each returned value.

See Also

HasConditions
HasTriggers
TriggerTrue
TriggerCount
SubTriggerCount
TriggerName
SubTriggerName
TriggerConfigured
TriggerBuildUI
TriggerProcessPostUI
TriggerFormatUI
Condition
TriggerReferencesDevice
TriggerFire
GetTriggers

Home > Actions

Actions

Articles in this section

See Also

Document Revisions
Getting Started
Controlling with JSON
Plugin Initialization
Base Plugin API
Devices
Callbacks
Triggers
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater
Appendices

Home > Actions > ActionCount

ActionCount

```
Public Function ActionCount() As Integer Implements HomeSeerAPI.IPlugInAPI.ActionCount
```

Return the number of actions the plugin supports.

Parameters: None

Returns: (Integer)

See Also

HandleAction
ActionFormatUI
ActionProcessPostUI
ActionBuildUI
ActionConfigured
GetActions
ActionName
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

Home > Actions > HandleAction

HandleAction

`Public Function HandleAction(ByVal ActInfo As IPlugInAPI.strTrigActInfo) As Boolean Implements HomeSeerAPI.IPlugInAPI.HandleAction`

When an event is triggered, this function is called to carry out the selected action. Use the ActInfo parameter to determine what action needs to be executed then execute this action.

Return TRUE if the action was executed successfully, else FALSE if there was an error.

Parameters:

ActInfo: (IPlugInAPI.strTrigActInfo)

Returns: True or False

See Also

ActionCount
ActionFormatUI
ActionProcessPostUI
ActionBuildUI
ActionConfigured
GetActions
ActionName
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

Home > Actions > ActionFormatUI

ActionFormatUI

`Public Function ActionFormatUI(ByVal ActInfo As IPlugInAPI.strTrigActInfo) As String Implements HomeSeerAPI.IPlugInAPI.ActionFormatUI`

Body of text here

See Also

ActionCount
HandleAction
ActionProcessPostUI
ActionBuildUI
ActionConfigured
GetActions
ActionName
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

Home > Actions > ActionProcessPostUI

ActionProcessPostUI

```
Public Function ActionProcessPostUI(PostData As Collections.Specialized.NameValueCollection, _
                                    ActInfoIN As IPlugInAPI.strTrigActInfo ) As IPlugInAPI.strMultiReturn _
                                    Implements HomeSeerAPI.IPlugInAPI.ActionProcessPostUI
```

When a user edits your event actions in the HomeSeer events, this function is called to process the selections.

Parameters:

PostData=(NameValueCollection) A collection of name value pairs that include the user's selections.

ActInfoIN (IPlugInAPI.strTrigActiInfo) Object that contains information about the action.

Returns: (IPlugInAPI.strMultiReturn) Object the holds the parsed information for the action. HomeSeer will save this information for you in the database.

Suppose your plug-in displays a checkbox in the HomeSeer actions for your plug-in. When this checkbox is checked or unchecked this function will be called. Look at the PostData name/value pairs and detect if the box was checked or unchecked. If it was checked, set the return value like:

```
Ret.TrigActInfo.SubTANumber = 1
```

If it was unchecked, set it like:

```
Ret.TrigActInfo.SubTANumber = 2
```

HomeSeer will then send you the value again when your ActionFormatUI function is called and you display the proper value.

See Also

ActionCount
HandleAction
ActionFormatUI
ActionBuildUI
ActionConfigured
GetActions
ActionName
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

ActionBuildUI

`Public Function ActionBuildUI(ByVal sUnique As String, ByVal ActInfo As IPlugInAPI.strTrigActInfo) As String Implements HomeSeerAPI.IPlugInAPI.ActionBuildUI`

This function is called from the HomeSeer event page when an event is in edit mode. Your plug-in needs to return HTML controls so the user can make action selections. Normally this is one of the HomeSeer jquery controls such as a clsJquery.jqueryCheckbox.

Parameters:

sUnique: (String) A unique string that can be used with your HTML controls to identify the control. All controls need to have a unique ID.

ActInfo: (IPlugInAPI.strTrigActInfo) Object that contains information about the action like current selections.

Returns: (String) HTML controls that need to be displayed so the user can select the action parameters.

See Also

ActionCount
HandleAction
ActionFormatUI
ActionProcessPostUI
ActionConfigured
GetActions
ActionName
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

ActionConfigured

`Public Function ActionConfigured(ByVal ActInfo As IPlugInAPI.strTrigActInfo) As Boolean Implements HomeSeerAPI.IPlugInAPI.ActionConfigured`

Return TRUE if the given action is configured properly. There may be times when a user can select invalid selections for the action and in this case you would return FALSE so HomeSeer will not allow the action to be saved.

Parameters:

ActInfo: (IPlugInAPI.strTrigActInfo) Object describing the action.

Returns: (TRUE or FALSE)

See Also

ActionCount
HandleAction
ActionFormatUI
ActionProcessPostUI
ActionBuildUI
GetActions
ActionName
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

GetActions

`Public Function GetActions(ByVal PIName As String) As HomeSeerAPI.IPlugInAPI.strTrigActInfo() Implements IAppCallbackAPI.GetActions`

TBD

See Also

ActionCount
HandleAction
ActionFormatUI
ActionProcessPostUI
ActionBuildUI
ActionConfigured
ActionName
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

ActionName

`Public ReadOnly Property ActionName(ByVal ActionNumber As Integer) As String Implements HomeSeerAPI.IPlugInAPI.ActionName`

Return the name of the action given an action number. The name of the action will be displayed in the HomeSeer events actions list.

Parameters:

ActionNumber: (Integer) The number of the action. Each action is numbered, starting at 1.

Example:

This example is for a plug-in that supports 2 actions:

```
    Public ReadOnly Property ActionName(ByVal ActionNumber As Integer) As String Implements
HomeSeerAPI.IPlugInAPI.ActionName
        Get
            If Not ValidAct(ActionNumber) Then Return ""
            Select Case ActionNumber
                Case 1
                    Return IFACE_NAME & ": Set Weight Option"
                Case 2
                    Return IFACE_NAME & ": Voltage Actions"
            End Select
            Return ""
        End Get
    End Property
```

See Also

ActionCount
HandleAction
ActionFormatUI
ActionProcessPostUI
ActionBuildUI
ActionConfigured
GetActions
ActionAdvancedMode
UpdatePlugAction
ActionReferencesDevice

ActionAdvancedMode

`Public Property ActionAdvancedMode As Boolean Implements HomeSeerAPI.IPlugInAPI.ActionAdvancedMode`

The HomeSeer events page has an option to set the editing mode to "Advanced Mode". This is typically used to enable options that may only be of interest to advanced users or programmers. The Set in this function is called when advanced mode is enabled. Your plug-in can also enable this mode if an advanced selection was saved and needs to be displayed.

Parameters:

Set: (Boolean) Set to TRUE to enable advanced mode, you should display advanced controls.

Get: (Boolean) Return TRUE if advanced mode is set, you may enable this mode if you detect advanced selections have already been made.

Example:

```
    Private mvarActionAdvanced As Boolean
    Public Property ActionAdvancedMode As Boolean Implements HomeSeerAPI.IPlugInAPI.ActionAdvancedMode
        Set(ByVal value As Boolean)
            mvarActionAdvanced = value
        End Set
        Get
            Return mvarActionAdvanced
        End Get
    End Property
```

See Also

ActionCount
HandleAction
ActionFormatUI
ActionProcessPostUI
ActionBuildUI
ActionConfigured
GetActions
ActionName
UpdatePlugAction
ActionReferencesDevice

Home > Actions > UpdatePlugAction

UpdatePlugAction

```
Function UpdatePlugAction(ByVal PlugName As String, ByVal evRef As Integer, ByVal ActionInfo As
IPlugInAPI.strTrigActInfo) As String
```

See Also

- ActionCount
- HandleAction
- ActionFormatUI
- ActionProcessPostUI
- ActionBuildUI
- ActionConfigured
- GetActions
- ActionName
- ActionAdvancedMode
- ActionReferencesDevice

Home > Actions > ActionReferencesDevice

ActionReferencesDevice

```
Public Function ActionReferencesDevice(ByVal ActInfo As IPlugInAPI.strTrigActInfo, _
                                       ByVal dvRef As Integer) As Boolean _
                                        Implements HomeSeerAPI.IPlugInAPI.ActionReferencesDevice
```

Return True if the given device is referenced in the given action.

Parameters:

ActInfo: (strTrigActInfo)

dvRef: (Integer) The device reference number to check.

Example:

```
Dim Act As MyAct = Nothing
Try
    Act = GetAction(ActInfo)
Catch ex As Exception
    Act = Nothing
End Try
If Act Is Nothing Then Return False
Return Act.Devices.Contains(dvRef)
```

See Also

Webpages

Articles in this section

The interface for a plugin is a webpage. In HS2 there were 2 functions that were called to handle web pages, GenPage and PutPage. These functions are still available but we highly recommend using our new PageBuilder class to build your web pages. By using this class you will have available a rich set of jquery controls to build your web pages with. Jquery will allow you to create

See Also

GetPagePlugin

```
Public Function GetPagePlugin(ByVal pageName As String, ByVal user As String, ByVal userRights As Integer,
ByVal queryString As String) As String Implements HomeSeerAPI.IPlugInAPI.GetPagePlugin
```

Web pages that use the clsPageBuilder class and registered with hs.RegisterLink and hs.RegisterConfigLink will then be called through this function. A complete page needs to be created and returned.

Parameters:
pageName: the name of the page as passed to the hs.RegisterLink function
user: name of logged in user
userRights: rights of logged in user

Returns: a complete HTML web page

See Also

PageBuilder (clsPageBuilder)
Page Performance
PostBackProc
GenPage
PagePut
RegisterConfigLink
RegisterHelpLink
RegisterLink
UnRegisterHelpLinks
ASP.NET

PageBuilder (clsPageBuilder)

The clsPageBuilder class is used to build your user interface in your plug-in. It provides an easy way to build web pages that support jquery. Jquery allows you to display robust HTML controls that allow page updating without page refreshes. This includes the ability to dynamically update pages without a refresh. See the sample plugin for some page examples.

The following sections describe the jquery controls and page builder functions that are available.

See Also

GetPagePlugin
Page Performance
PostBackProc
GenPage
PagePut
RegisterConfigLink
RegisterHelpLink
RegisterLink
UnRegisterHelpLinks
ASP.NET

Building a Page

The clsPageBuilder class is used to assist in building web pages. It is also required for creating pages for use with the HomeSeer registerLink API. See the sample plug-in WebPage.vb for a sample page.

See Also

reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormStart
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

reset

Called from the start of your page to reset all internal data structures in the clsPageBuilder class, such as menus.

For example:

```
Public Function GetPagePlugin(ByVal pageName As String, ByVal user As String, ByVal userRights As Integer,
ByVal queryString As String) As String
        Dim stb As New StringBuilder

        Me.reset()
 stb.Append("This is the page text")
        Me.AddBody(stb.ToString)
        Return Me.BuildPage()
End Function
```

See Also

- Building a Page
- AddHeader
- AddBody
- AddFooter
- suppressDefaultFooter
- BuildPage
- DivStart
- DivEnd
- FormStart
- FormEnd
- divToUpdate
- pageCommands
- JQuery Controls (clsJquery)
- AddAjaxHandlerPost

Home > Webpages > PageBuilder (clsPageBuilder) > AddHeader

AddHeader

Adds the standard HomeSeer header to your web page which includes the top header bar and menus. A call to hs.GetPageHeader needs to be called first to get the actual header from HomeSeer.

For example:

```
Public Function GetPagePlugin(ByVal pageName As String, ByVal user As String, ByVal userRights As Integer,
ByVal queryString As String) As String
        Dim stb As New StringBuilder

        Me.reset()
        Me.AddHeader(hs.GetPageHeader(pageName, "Sample Plugin", "", "", False, True))
 stb.Append("This is the page text")
        Me.AddBody(stb.ToString)
        Return Me.BuildPage()
End Function
```

See Also

- Building a Page
- reset
- AddBody
- AddFooter
- suppressDefaultFooter
- BuildPage
- DivStart
- DivEnd
- FormStart
- FormEnd
- divToUpdate
- pageCommands
- JQuery Controls (clsJquery)
- AddAjaxHandlerPost

AddBody

Adds the body of the page to the clsPageBuilder class. The body is any html that you want to display.

For example:

```
Public Function GetPagePlugin(ByVal pageName As String, ByVal user As String, ByVal userRights As Integer,
ByVal queryString As String) As String
        Dim stb As New StringBuilder


        Me.reset()
        Me.AddHeader(hs.GetPageHeader(pageName, "Sample Plugin", "", "", False, True))
 stb.Append("This is the page text")
        Me.AddBody(stb.ToString)
        Return Me.BuildPage()
End Function
```

See Also

- Building a Page
- reset
- AddHeader
- AddFooter
- suppressDefaultFooter
- BuildPage
- DivStart
- DivEnd
- FormStart
- FormEnd
- divToUpdate
- pageCommands
- JQuery Controls (clsJquery)
- AddAjaxHandlerPost

AddFooter

Adds the default footer to the page. This is the standard footer that appears on all HomeSeer web pages. This is optional. Since the clsPageBuilder class in your plugin is the actual class used in HomeSeer, you need to suppress the default footer and then add the HomeSeer footer by calling hs.GetPageFooter.

For example:

```
Public Function GetPagePlugin(ByVal pageName As String, ByVal user As String, ByVal userRights As Integer,
ByVal queryString As String) As String
        Dim stb As New StringBuilder

        Me.reset()
        Me.AddHeader(hs.GetPageHeader(pageName, "Sample Plugin", "", "", False, True))
 stb.Append("This is the page text")
        Me.AddBody(stb.ToString)
        Me.AddFooter(hs.GetPageFooter)
        Me.suppressDefaultFooter = True
        Return Me.BuildPage()
End Function
```

See Also

Building a Page
reset
AddHeader
AddBody
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormStart
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

Home > Webpages > PageBuilder (clsPageBuilder) > suppressDefaultFooter

suppressDefaultFooter

Suppresses the default page footer. This is required to be true for plug-ins. Call HS.GetPageFooter to get the real page footer that HomeSeer uses.

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
BuildPage
DivStart
DivEnd
FormStart
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

Home > Webpages > PageBuilder (clsPageBuilder) > BuildPage

BuildPage

This is the last function that you need to call to actually have clsPageBuilder build your complete page. It returns the full HTML of your page.

For example:

```
Public Function GetPagePlugin(ByVal pageName As String, ByVal user As String, ByVal userRights As Integer,
ByVal queryString As String) As String
        Dim stb As New StringBuilder

        Me.reset()
        Me.AddHeader(hs.GetPageHeader(pageName, "Sample Plugin", "", "", False, True))
 stb.Append("This is the page text")
        Me.AddBody(stb.ToString)
        Me.AddFooter(hs.GetPageFooter)
        Me.suppressDefaultFooter = True
        Return Me.BuildPage()
End Function
```

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
DivStart
DivEnd
FormStart
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

DivStart

```
Public Shared Function DivStart(ByVal id As String, ByVal attribute As String) As String
```

Returns a formatted html DIV. Pass the ID and attribute you want to use.

For example:

The following creates a new DIV named "errormessage" and applies the class named "errormessage" to it:

```
stb.Append(clsPageBuilder.DivStart("errormessage", "class='errormessage'"))
```

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivEnd
FormStart
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

DivEnd

Ends a DIV that was started with DivStart.

For example:

```
stb.Append(clsPageBuilder.DivStart("errormessage", "class='errormessage'"))
stb.Append(clsPageBuilder.DivEnd)
```

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
FormStart
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

FormStart

```
FormStart("form_name", "page_name", "post")
```

Returns HTML that is a start of a form. Use FormEnd to end the form.

Parameters:

form_name: Name of the form

page_name: Name of the page the form is on

post: Method of posting the form, nomally this is the string "post".

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

FormEnd

Ends a form. There are no parameters.

Returns HTML that is the form end tag.

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormStart
divToUpdate
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

Home > Webpages > PageBuilder (clsPageBuilder) > divToUpdate

divToUpdate

When your postback function is called, you may want to dynamically update some content on the page. If the content is enclosed in a DIV tag and assigned an ID, you can update the HTML in that div. This function takes 2 parameters:

divID: The ID of the DIV to update
divHTML: The HTML text that should replace the current HTML at the specified DIV

For example:

In your GetPagePlugin function you have a statement that displays the current time:

```
stb.Append("<div id='current_time'>" & DateTime.Now.ToString & "</div>" & vbCrLf)
```

Now you want to update this time from a timer. The timer will post to your postBackProc at a specific interval. To update the DIV named "current_time", you can do this:

```
Me.divToUpdate.Add("current_time", DateTime.Now.ToString)
```

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormStart
FormEnd
pageCommands
JQuery Controls (clsJquery)
AddAjaxHandlerPost

Home > Webpages > PageBuilder (clsPageBuilder) > pageCommands

pageCommands

In your postBackProc you can update DIV tags using divToUpdate. There are also some page commands that update other items on the page, or execute javascript that perform specific functions, such as re-directing to another page.

There are 2 different types of commands, page commands, and property set commands. The page commands normally relate to operations on a page. The property set functions deal with individual elements on a page such as divs and control items.

Command	**Parameter**	**Description**
Me.pageCommands.Add("newpage", url)	url	Displays a new page using the given URL

Me.pageCommands.Add("popmessage", msg)	message string	Displays a pop-up message box using the javascript alert() function
Me.pageCommands.Add("starttimer", "")	none	Starts a timer. A timer is started with the command page.RefreshIntervalMilliSeconds=value
Me.pageCommands.Add("stoptimer", "")	none	Stops a timer
Me.pageCommands.Add("opendialog", dialog_div)	dialog div	Opens a jquery dialog
Me.pageCommands.Add("closedialog", dialog_div)	dialog div	Closes a jquey dialog
Me.pageCommands.Add("refresh", "true")	must be set to "true"	Causes the current page to refresh
Me.pageCommands.Add("delayhide", dialog_div)	dialog div	Delays the hiding of a dialog based on the timer setting, IE: Me.pageCommands.Add("timerdelay", "4000")
Me.pageCommands.Add("slidingtabopen", sliding_tab_id)	id of sliding tab	Expands a given sliding tab (sliding tab is created with clsJQuery.jqSlidingTab)
Me.pageCommands.Add("slidingtabclose", sliding_tab_id)	id of sliding tab	Closes a given sliding tab
Me.pageCommands.Add("stopspinner",spin_name)	id of spinner	Stops a spinner created with jqSpinner
Property Set Commands	**Parameters**	**Description**
Me.propertySet.Add(id,"removeattr=ATTR")	id=id of element, ATTR=attribute to remove	removes an attribute from an element. For example, to remove the "disabled" attribute from a checkbox with the id of "mycheck": Me.propertySet.Add("mycheck","removeattr=disabled"
Me.propertySet.Add(id,"addattr=ATTR")	id=id of element, ATTR=attribute to remove	Same as removeattr above but adds an attribute
Me.propertySet.Add(id,"setcss=CSS=VALUE")	id=id of element, CSS=VALUE=css to change	Adds CSS style attributes to a given element. For example, to make a DIV hidden using CSS, assuming the id of the element is "mydiv": Me.propertySet.Add("mydiv","setcss=visibility=hidden")
Misc Commands		
executefunction	jquery command to execute	During postback it may be desirable to execute a jquery or javascript function. The command takes javascript as a parameter and executes that code during a postback. For example, if you have a button named "mybutton" and you want to force a click on this button the command would be: Me.pageCommands.Add("executefunction", "$('#" & "mybutton" & "').click();") Note: You can only add one function per page command. If you need to call multiple functions, use multiple page commands.

Note:

HomeSeer appends an identifier to all jquery elements it adds to the page. This identifer is named "clsJQuery.ID_IDENT". This is appended to the element's ID. It is stripped off before the postback is called so normally is of no concern. However, if you are modifying elelments with the above commands, you need to be aware of this identifier and append it to any ID's you pass. Note that ID_IDENT is only appended if an ID is not explicitly set when a jquery control is created. If you DO specify an ID it is not appended.

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormStart
FormEnd
divToUpdate
JQuery Controls (clsJquery)
AddAjaxHandlerPost

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery)

JQuery Controls (clsJquery)

This section lists the available jquery controls that you can use on your web pages. This includes plug-in configuration and control pages, as well device configuration pages. These controls will only work on pages that are built using the clsPageBuilder class. The exception is the device configuration pages, as this page is built using the clsPageBuilder class so you only need to return HTML for your portion of the page. See ConfigDevice for more information.

The following properties are available for all controls:

Public Properties:

Property	Type	Default	Description
toolTip	String	""	When the mouse hovers over the control, this tooltip message will display
enabled	Boolean	True	Set to False to disable the control. It will be displayed, but the user will not be able to interact with it
className	String	""	Set this to apply a class to the control, normally from a stylesheet
left,top,width,height	Integer	auto	Set these values to adjust the position and size of a control
visible	Boolean	True	False=control is not visible
style	String	""	Sets style attributes to the control. If multiple styles are to be added, use AddStyle

Public Methods:

Method	Parameters	Description
AddStyle	String p_style	Adds a style to the control

Example:

The following example adds a tooltip to a button:

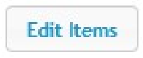
```
Dim b1 As New clsJQuery.jqButton("b1", "Hide", "test", False)
b1.toolTip = "This button will hide the slider"
stb.Append(b1.Build)
```

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormStart
FormEnd
divToUpdate
pageCommands
AddAjaxHandlerPost

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqButton

jqButton



Displays a simple button. When pressed, the ID of the button is submitted. Optionally, any form that the button is in also is submitted.

Properties:

Property	Type	Default	Description
primaryIcon	String	""	Path to the icon to display, the icon is displayed on the standard button
secondayIcon	String	""	Path to the icon to display, the icon is displayed on the standard button
stayOnPage	Boolean	True	True=do not refresh the page
imagePathNormal	String	""	If present, the button is converted to an image
imagePathPressed	String	""	If present, the image is displayed when the button is pressed, imagePathNormal must be set
imagePathHover	String	""	If present, the image is displayed when the mouse hovers over the image
hyperlink	Boolean	False	If True, the button is formatted as underlined text
includeClickFunction	Boolean	True	Uses a javascript function to handle clicks, else posts to the form
functionToCallOnClick	String	""	If a custom javascript is needed to be called on click, enter the function name. You must provide the function
url	String	""	URLl of a web page to open when button is clicked, the page is opened in the same window unless the urlNewWindow property is set to True.
urlNewWindow	Boolean	False	True=open URL in a new window

Creation:

`New(ByVal p_name As String, ByVal p_label As String, ByVal p_page As String, ByVal p_submit_form As Boolean)`

Parameters	Type	Description
p_name	String	Identifier for control
p_label	String	Text label for the button (if not image button)
p_page	String	Name of the page this control is on
p_submit_form	Boolean	True=submit all the controls in the contained form

Example:

This example displays a textbox and a button in a form and submits the form when the button is clicked.

```
    stb.Append(clsPageBuilder.FormStart("myform2", "testpage", "post"))
    stb.Append(clsPageBuilder.TextBox("name2", "name2", "text", "", 8))
    Dim b2 As New clsJQuery.jqButton("button1", "Button for form2", "test", True)
    stb.Append(b2.Build)
    stb.Append(clsPageBuilder.FormEnd)
```

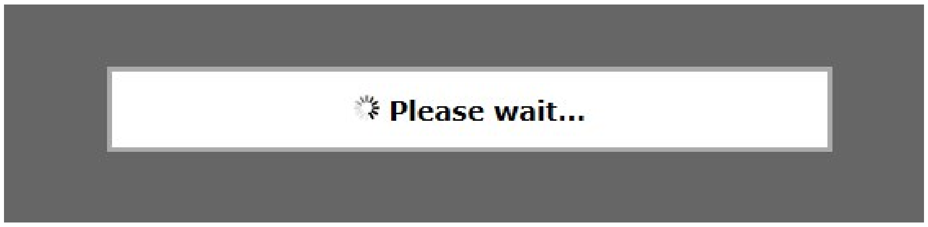
Here is a sample postback procedure that detects the button press:

```
Public Overrides Function postBackProc(ByRef state As StateObject, ByVal Data As String) As String
     Dim parts As Collections.Specialized.NameValueCollection
     parts = HttpUtility.ParseQueryString(Data)
     If parts("id") = "button1" then
        ' do something
     End If
End Sub
```

See Also

jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

jqBlockUI



Blocks out the screen so no user input is allowed and displays a given text message. After the message is displayed, the given URL is called with a GET and any long running task can be executed. When the task completes, update the screen DIV with me.DivToUpdate(...).

Creation:

New(ByVal p_url_source As String, ByVal p_div_to_update As String, ByVal p_message As String)

Parameters	Type	Description
p_url	String	The URL to GET after the message is displayed
p_div_to_update	String	The DIV that will be updated with the data from the GET call
p_message	String	The message to display

Example:

This example displays the block message then updates the given DIV with the HTML from the GET call.

```
   ' Add the blockUI control and give it a URL to call after the page displays, the URL here is "PageTemplate?
realpage=yes", the
   ' DIV that the control will update is named "PageTemplage" (set in the line above)
   Dim bui As New clsJQuery.jqBlockUI("PageTemplate?realpage=yes", "PageTemplate", "")
   stb.Append(bui.Build)
```

See Also

jqButton
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqCheckBox

jqCheckBox

Displays a check box. When the checkbox is selected or unselected a postback is performed and the ID of the control is posted along with the value. Optionally, the form containing the control may be posted.

Properties:

Property	Type	Default	Description
checked	Boolean	False	If True, the checkbox is checked
functionToCallOnClick	String	""	When the control is clicked, this javascript function will be called. You must supply the function
sliderStyle	Boolean	False	True=Use slider style as opposed to checkbox

Creation:

`New(ByVal p_name As String, ByVal p_label As String, ByVal p_page As String, ByVal p_autoPostBack As Boolean, ByVal p_submit_form As Boolean)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_label	String	Text label for the checkbox
p_page	String	Name of the page this control is on
p_autoPostBack	Boolean	Normally set to True, if False, the control will not perform a postback when the control is clicked
p_submit_form	Boolean	True=submit all the controls in the contained form

Different combinations of Autopostback and SubmitForm will produce different results.

AutoPostback	SubmitForm	Result
True	True	The object will post and will submit the form it's in. Values will be 'checked' and 'unchecked'
True	False	The object will post. Values will be 'checked' and 'unchecked'
False	True	The object will post when the form it's in is submitted. Values will be 'checked' and 'unchecked'
False	False	The object will post when the form it's in is submitted. Values will be 'on' and ''

Example:

This example displays a checkbox:

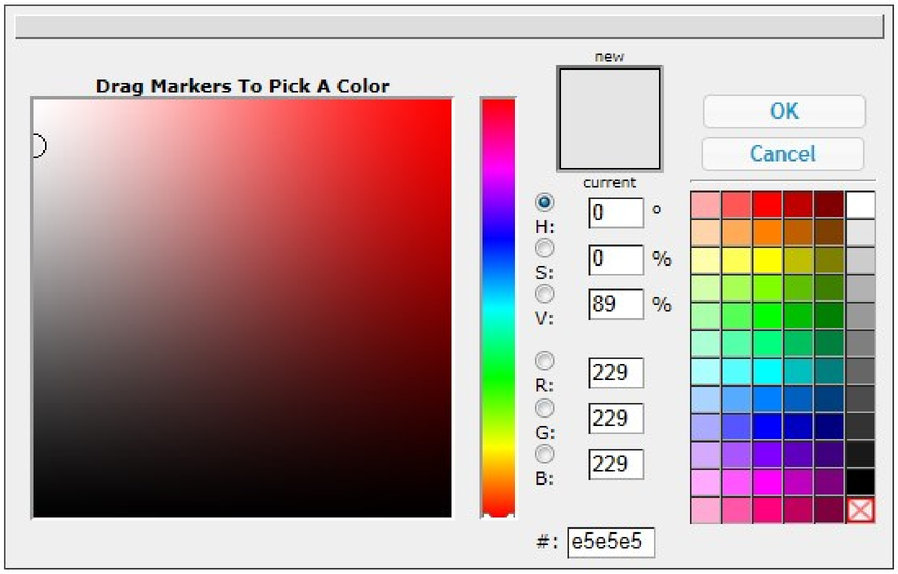
```
        Dim c As New clsJQuery.jqCheckBox("check1", "CheckLabel", "test", True, False)
        c.checked = True
        stb.Append(c.Build)
```

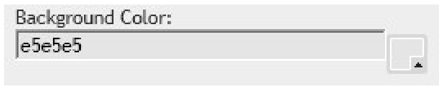
See Also

- jqButton
- jqBlockUI
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqListBoxEx
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqColorPicker

jqColorPicker




Background Color:
e5e5e5

Displays the current color choice in a read-only textbox and allows the user to display a color picker in order to choose a color.

Creation:

`New(ByVal p_name As String, ByVal p_page As String, ByVal p_size as Integer, ByVal p_color as String)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Name of the page this control is on
p_size	Integer	Size of input box that displays the current color
p_color	String	The color to display in the input box and in the color picker by default

Example:

```
Dim cp As New clsJQuery.jqColorPicker("color", "status", 40, "0000")
       stb.Append(cp.Build)
```

See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqListBoxEx
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

jqContainer

Displays a container for other controls and text.

Properties:

Property	Type	Default	Description
clickNameValue	String	""	When the container is clicked, a post back will occur if this property is set. The name value passed to the post back is this string, like action=new_level
backgroundImage	String	""	The image to be used as the background of the container
backgroundColor	String	"transparent"	The color to be used as the container background
stretchBackgroundImage	Boolean	False	True=Stretch background image to fill container
contextMenuPostData	String	""	Data to post is mouse is right clicked on the container
positionAbsolute	Boolean	True	If true, positioning is relative the page, if false, positioning is relative to the containing DIV

Creation:

```
New(ByVal p_name As String, ByVal p_parentID As String, ByVal p_title As String, ByVal p_title_image As
```

String, ByVal p_top As Integer, ByVal p_left As Integer, ByVal p_width As Integer, ByVal p_height As Integer, ByVal p_content As String, ByVal p_nested As Boolean, ByVal p_page As String, ByVal p_edit As Boolean)

Parameters	Type	Description
p_name	String	Identifier for the control
p_parentID	String	If containers are nested, this is the ID of the parent container
p_title	String	Title to be displayed, if an empty string, not title will be displayed
p_title_image	String	Image to be displayed in the title
p_top,p_left,p_width,p_height	Integer	Location and size of the container
p_content	String	The HTML content to display in the container
p_nested	Boolean	True=The container is nested inside another container
p_page	String	The page name the container is on
p_edit	Boolean	True=Container is in edit mode and it may be re-sized and moved by the user

Example:

This example displays a single container with a title. It creates a DIV and adds the container to this div with relative positioning. It also sets a custom background color on the container.

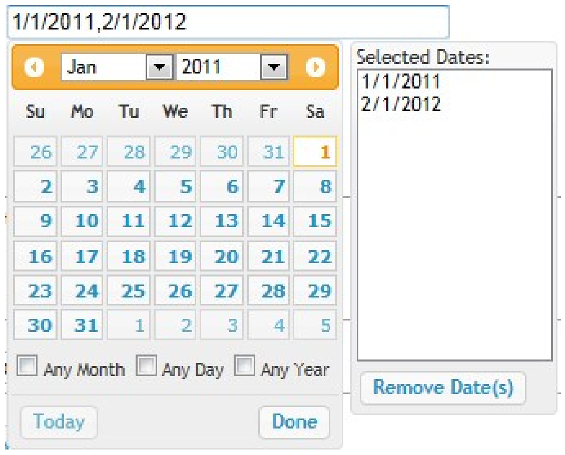
```
        stb.Append(clsJQuery.DivStart("container_div", "", False, False, "", "", "test"))
        Dim room As New clsJQuery.jqContainer("parent1", "", "Container Title", "", 0, 0, 200, 100, "This is
the container content on the first line<br><b>This is the second line in bold</b>", False, "test", False)
        room.backgroundColor = "e5e5e5"
        room.positionAbsolute = False

        stb.Append(room.build)
        stb.Append(clsJQuery.DivEnd)

        stb.Append(clsPageBuilder.DivStart("", "id='result'"))
        stb.Append(clsPageBuilder.DivEnd)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqDatePicker

jqDatePicker



Displays a dialog for choosing a specific date. Supports selection of multiple dates.

Properties:

Property	Type	Default	Description
size	Integer	8	The size of the read-only textbox that displays the selected date(s)
multipleDates	Boolean	False	If True, multiple dates may be selected, else only a singe date
allowWildCards	Boolean	False	True=allow wildcards in the dates such as 1/12/* to select any year
defaultDate	String	current date	Date to pre-select in the dialog

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_label As String, ByVal p_page As String, ByVal p_submit_form As Boolean, ByVal p_post_seperator As String)`

Parameters	Type	Description
p_name	String	Name attribute to assign the controll
p_label	String	Text to display next to the read-only textbox
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected
p_post_seperator	Integer	Seperator character to use when posting multiple dates

Example:

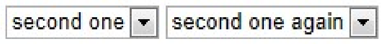
```
Dim dp As New clsJQuery.jqDatePicker("mydp", "Date:", "test", False, ",")
dp.multipleDates = True
dp.allowWildcards = True
dp.defaultDate = "1/1/2011,2/1/2012"
dp.size = 40
stb.Append(dp.Build)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqDropList

jqDropList



Displays a list of items that may be selected.

Properties:

Property	Type	Default	Description
items	Collections.Generic.List(Of pair)	Empty	Collection of name/value pairs that represent a selectable item (use AddItem to add items)
selectedItemIndex	Integer	0	The index of the selected item to display by default
autoPostBack	Boolean	True	If True, when an item is selected, a post back occurs with the selection

Methods:

Property	Parameters	Description

AddItem	p_name = item name
	p_value = item value
	p_selected = if True, item is selected
		Adds a name/value pair to the list

ClearItems	None	Clears all the items in the list

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_page As String, ByVal p_submit_form As Boolean)`

Parameters	Type	Description
p_name	String	Name attribute to assign the controll
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected

Example:

```
Dim dl As New clsJQuery.jqDropList("droplistname", "test", True)
dl.AddItem("first one", "1", False)
dl.AddItem("second one", "2", True)
```

```
        stb.Append(dl.Build)
```

See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqListBoxEx
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

jqDynSpinner

Displays a spinner graphic. The spinner is a dynamic spinner and may be customized when created. Not graphic is used.

Properties:

Property	Type	Default	Description
top	String	"auto"	Location top
left	String	"auto"	Location left
lines	Integer	12	Number of lines in the spinner
length	Integer	7	Length of the spinner
width	Integer	2	Width of the spinner
radius	Integer	5	Radius of the spinner
color	String	"#000"	Spinner color
speed	Integer	1	Spinning speed
trail	Integer	66	Size of trail
shadow	Boolean	False	True=display a shadow under spinner

Creation:

`Public Sub New(ByVal p_elementid As String)`

Parameters	Type	Description
p_elementid	String	ID of the spinnerl

Example:

Display a spinner at the current location on the page (relative to other HTML):

```
        Dim sp As New clsJQuery.jqDynSpinner("spin")
        sp.left = 0
        stb.Append(sp.Build)
```

In the post back handler, the spinner may be disabled with the call:

```
        Me.pageCommands.Add("stopspinner", "spin")
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqFileUploader

jqFileUploader

Displays a file selection box for selecting a file on the user's local file system and uploads the file.

Properties:

Property	Type	Default	Description
values	Collections.Specialized.NameValueCollection	Empty	Collection of name/value pairs that will be posted with the upload
extensions	Collections.Specialized.NameValueCollection	Empty	Collection of allowable extensions like *.jpg
acceptFiles	String	""	File types like "audio/*"

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_page As String)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on

Example:

```
        Dim ul As New clsJQuery.jqFileUploader("uploader", "test")
        ul.values.Add("uploadfile", "true")
        ul.AddExtension("*.jpg")
        ul.AddExtension("*.wav")
        ul.AddExtension("*.png")
        ul.acceptFiles = "audio/*"
        ul.label = "Upload File..."
        stb.Append(ul.Build)
```

When the user selects a file, it is uploaded to a temp file and then a post back is done to your page. In the post back you can handle processing the file. The post back can handled as follows:

```
        If parts("uploadfile") = "true" Then
             ' handle uploading of file
             ' the orig file is ID_OriginalFile
             ' the temp file is ID_TempFile
             dim tempFile as String = parts("ID_TempFile")
             ' process the temp file here

             ' status is ID_Status
             ' return the proper JSON so the uploader completes
             ' success return=Return "{""success"":""true""}"
             ' if error return=Return "{""error"":""No directory specified""}"
             Return "{""success"":""true""}"
        End If
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqListBox

jqListBox

Displays a list of items that may be selected. The style of the control needs to be set using the common style propertly. See the example below.

This version of the listbox returns the name selected. For a version of this control that handles name/value pairs, use the jqListBoxEx.

Note that the returned selection is URLEncoded for cases where the value in the list box may contain an "&". In your postback, gather all name/value pairs and URLDecode the returned value. An example postback handler might be:

```
      Dim parts As Collections.Specialized.NameValueCollection
      parts = HttpUtility.ParseQueryString(Data)
      dim selection as string = HTTPUtility.URLDecode(parts("list1"))
```

Properties:

Property	Type	Default	Description
items	Collections.Generic.List(Of pair)	Empty	Collection of name/value pairs that represent a selectable item. Note that the value is ignored in this control. See jqListBoxEx.

Creation:

`Public Sub New(p_name As String, ByVal p_page As String)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on

Example:

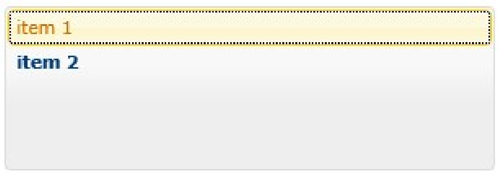
```
Dim lb As New clsJQuery.jqListBox("list1", "test")
lb.style = "height:100px;width:300px;"
lb.items.Add("item 1")
lb.items.Add("item 2")
stb.Append(lb.Build)
```

See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBoxEx
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqListBoxEx

jqListBoxEx



Displays a list of items that may be selected. The style of the control needs to be set using the common style propertly. See the example below.

This version of the listbox control allows for name/values to be added. When an item is clicked, the value for the selection is returned.

On postback, the ID of the control as well as the value of the item clicked is returned. An example postback for a single click on an item might be:

```
id=list2&list2=3
```

If the item is double clicked, an extra parameter is returned. Note that for a double click, there is still a postback for each single click:

```
id=list2&list2=3&click=double
```

Properties:

Property	Type	Default	Description
items	Collections.Generic.List(Of pair)	Empty	Collection of name/value pairs that represent a selectable item
height	Integer	Empty	Sets a style parameter in pixels

Methods:

Methods	Parameters	Description

AddItem	p_name = Name of item
	p_value = Value of item
	p_selected = True if item is to be selected when the control is initially drawn
		Adds a name/value pair to the list.

Creation:

`Public Sub New(p_name As String, ByVal p_page As String)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on

Example:

```
Dim lb As New clsJQuery.jqListBoxEx("list1", "test")
lb.style = "height:100px;width:300px;"
lb.AddItem("item 1",1)
lb.AddItem("item 2",2)
stb.Append(lb.Build)
```

See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqLocalFileSelector

jqLocalFileSelector

Displays a dialog where a user can select a local file name. The file is NOT uploaded.Use the common label property to set the text to display on the default button. The default text is "Edit".

Properties:

Property	Type	Default	Description
modal	Boolean	False	If True, access to the page is restricted to the file dialog
path	String	"\"	The default directory to display files from
dialogCaption	String	""Select File"	Caption to display at the top of the dialog

Methods:

Method	Parameters	Description
AddExtension	None	Adds a valid extension like "*.*" or "*.jpg"

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_page As String, ByVal p_submit_form As String)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected

Example:

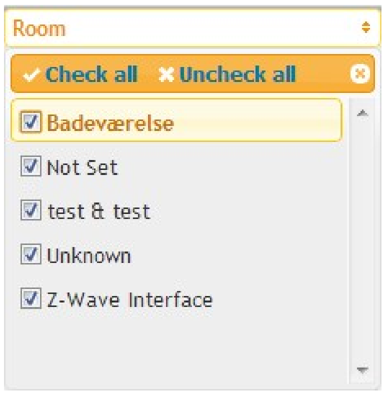
```
Dim fs As New clsJQuery.jqLocalFileSelector("fs1", "test", True)
fs.label = "My Folders"
fs.path = hs.GetAppPath
fs.AddExtension("*.*")
stb.Append(fs.Build)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqMultiSelect

jqMultiSelect



Similar to a standard drop list but allows for the selection of individual items using a checkbox. When any item is checked/unchecked a postpack occurs. A postback also occurs when the list is closed. When closed the postback includes all selected items.

Note that the returned selection is URLEncoded for cases where the value in the list box may contain an "&". In your postback, gather all name/value pairs and URLDecode the returned value. An example postback handler might be:

```
      Dim parts As Collections.Specialized.NameValueCollection
      parts = HttpUtility.ParseQueryString(Data)
      dim selection as string = HTTPUtility.URLDecode(parts("list1"))
```

Properties:

Property	Type	Default	Description
items	Collections.Generic.List(of pairsel)	Empty	Collection of name/value pairs that represent a selectable item. This may be used to view items that have been added. Use the method AddItem to actually add items

Methods:

Method	**Parameters**	**Description**
AddItem	p_name as String p_value as String p_selected as Boolean	Add an item that will appear in the list. Items added are in two parts, and name that will appear to the user, and a value that will be submitted when the item is selected. If p_selected is True, the item will be checked.

Creation:

`Public Sub New(p_name As String, ByVal p_page As String, p_submit_form)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the list will be submitted if it appears in a form. If False, the entire list is submitted when it is closed, and individual items are submitted when checked/unchecked.

Example:

```
       Dim msel As New clsJQuery.jqMultiSelect("msel1", "test", False)
msel.label = "Select test value"
msel.AddItem("Option Name 1", "val1", False)
msel.AddItem("Option Name 2", "val2", True)
msel.AddItem("Option Name 3", "val3", True)
stb.Append(msel.Build)
```

When a postback occurs, the postback data will be as follows. This example shows 2 items selected, note that the data is URL Encoded. Items are seperated with a "|".

```
msel1.close=val1%7Cval2
```

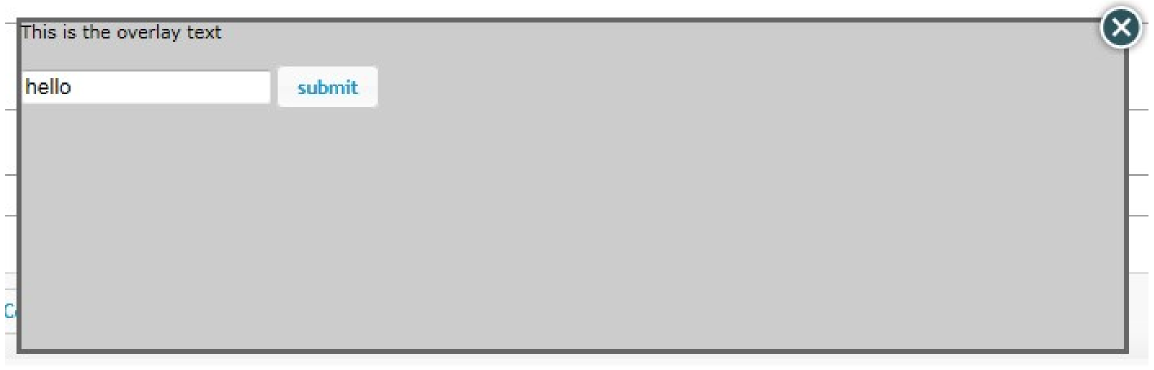
When one item is checked, the postback will be:

```
msel1_val2=checked
```

See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqListBoxEx
- jqLocalFileSelector
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

jqOverlay



Displays a pop-up overlay window that contains HTML contents. Note that the overlay cannot scroll so it should only be used with minimal content.

Properties:

Property	Type	Default	Description
overlayHTML	String	""	HTML that is the contents of the overlay

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_page As String, ByVal p_modal As Boolean, ByVal p_overlay_class As String)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on
p_modal	Boolean	True=user is locked out of entire page other than the overlay
p_overlay_class	String	Class to use to format the overlay, if no class is specified a default is created

Example:

```
Dim ol As clsJQuery.jqOverlay = New clsJQuery.jqOverlay("ov1", "test", False, "events_overlay")
ol.toolTip = "This will display the overlay"
ol.label = "Display Overlay"
Dim tbov1 As New clsJQuery.jqTextBox("tbov1", "text", "hello", "test", 20, False)
Dim tbut1 As New clsJQuery.jqButton("tbut1", "submit", "test", True)
ol.overlayHTML = "<div>This is the overlay text<br><br>" & tbov1.Build & tbut1.Build & "</div>"
stb.Append(ol.Build)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

jqProgressBar

Displays a progress bar that can show progress of a task. The progress can be updated in the postback handler.

Creation:

`Public Sub New(p_name As String, p_initial_value As Integer)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_initial_value	Integer	Initial value the progress bar is set to

Example:

Use the "className" property to assign a class to the control to specify its size, etc.

```
Dim pb As New clsJQuery.jqProgressBar("pb1", 50)
pb.className = "update_progress"
stb.Append(pb.Build)
```

When a post back occurs, normally from a timer, the value of the control can be updated as follows:

```
Me.propertySet.Add("pb1", "progressbar=" & ProgressBarValue.ToString)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

jqRadioButton

Displays radio like controls where only one can be enabled at a time

Properties:

Property	Type	Default	Description
values	Collections.Specialized.NameValueCollection	Empty	Collection of name/value pairs that represent a selectable item
buttonset	Boolean	True	If true, controls show as buttons, if false, they show as standard radio buttons

Creation:

Public Sub New(ByVal p_name As String, ByVal p_page As String, ByVal p_submit_form As Boolean)

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected

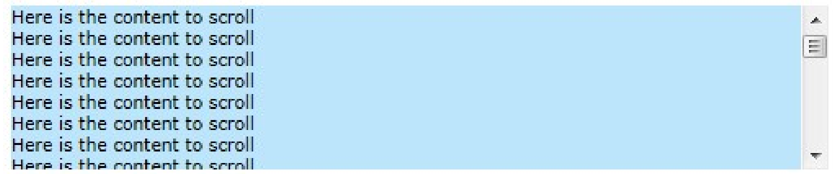
Example:

```
Dim rb As New clsJQuery.jqRadioButton("rb1", "test", False)
rb.values.Add("Item 1", "1")
rb.values.Add("Item 2", "2")
stb.Append(rb.Build)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

jqScrollingRegion


Here is the content to scroll
Here is the content to scroll
Here is the content to scroll
Here is the content to scroll
Here is the content to scroll
Here is the content to scroll
Here is the content to scroll
Here is the content to scroll

Displays an area that can contain content that will scroll. Content added to the region will be added to the bottom and force the current content to be scrolled. By default, the contol uses the class names "scroll-pane". You can use the style property to assign your own style, see the example below.

Properties:

Property	Type	Default	Description
content	String	""	The content to display in the region

Creation:

`Public Sub New(p_name As String)`

Parameters	Type	Description
p_name	String	Identifier for the control

Example:

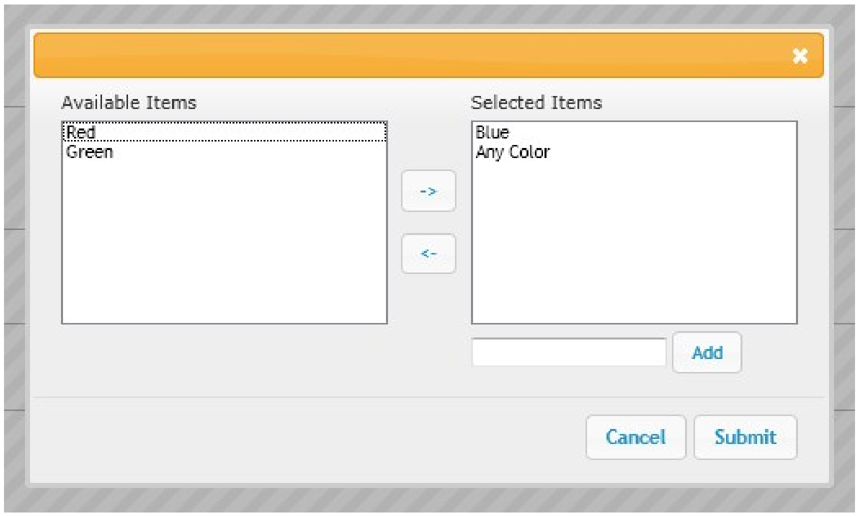
```
        Dim sr As New clsJQuery.jqScrollingRegion("my_region")
        sr.className = ""  ' do not use the default class, we will define the style here
        sr.AddStyle("height:100px;overflow:auto;width: 500px;background: #BCE5FC;") ' style to use, or set
classname to a class
        sr.content = "Here is the content to scroll<br>Here is the content to scroll"
        stb.Append(sr.Build)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqSelector

jqSelector



Displays a modal dialog allowing the user to select and unselect items.

Properties:

Property	Type	Default	Description
items	Collections.Generic.List(Of Pair)	Empty	Collection of name/value pairs that will be posted with the upload
selectedItems	Collections.Generic.List(Of Pair)	Empty	Collection of name/value pairs of items that are selected when the control is first displayed (show in right box)
size	Integer	40	Size of texbox that shows selections if defaultDisplayUseButton is false
defaultDisplayUseButton	Boolean	True	Displays a button that the use can press to display the dialog, else a textbox is displayed that shows the selections
formToPostID	String	""	The form ID to post to if submitting the fom. If empty then the closest form is used
dialogCaption	String	"Edit Selections"	The caption on the dialog

labelLeft	String	"Available Items"	The catption for the available items list box
labelRight	String	"Selected Items"	The caption for the selected items listbox
listSeperatorChar	String	","	Character to use to seperate selected items when posted
allowAddingMoreSelections	String	True	If false, the textbox to manually add selections is not shown

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_page As String, ByVal p_submit_form As String)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on

Example:

```
        Dim s As New clsJQuery.jqSelector("sel1", "test", "")
        s.AddItem("Red", "1", False)
        s.AddItem("Blue", "2", True)
        s.AddItem("Green", "3", False)
        s.AddItem("Any Color", "4", True)
        s.label = "Edit Colors"    ' label on button
 s.dialogCaption = "Select Colors"
        stb.Append(s.Build)
```

See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqListBoxEx
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

jqSlider

Displays a sliding control to select a range.

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_rangeStart As Integer, ByVal p_rangeEnd As Integer, ByVal p_value As Integer, ByVal p_orientation As jqSliderOrientation, ByVal p_height_width As Integer, ByVal p_page`

```
As String, ByVal p_submit_form As String)
```

Parameters	Type	Description
p_name	String	Identifier for the control
p_rangeStart	Integer	Start value for the control
p_rangeeEnd	Integer	End value for the control
p_value	Integer	Initial value for the control
p_orientation	jqSliderOrientation	Orientation for the control, value is either jqSliderOrientation.horizontal or jqSliderOrientation.verticall
p_height_width	Integer	Height or width of the controls (depends on orientation)
p_page	String	Page the control is on

Example:

```
        Dim slid As New clsJQuery.jqSlider("sliderid", 0, 100, 25,
clsJQuery.jqSlider.jqSliderOrientation.horizontal, 90, "status", "")
        stb.Append(slid.build)
```

On postback, the data posted will be:

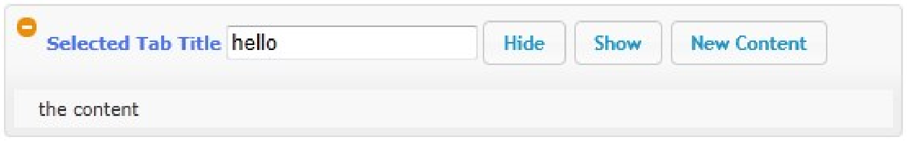
```
id=sliderid&value=54
```

See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqListBoxEx
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimePicker
- jqTimeSpanPicker
- jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqSlidingTab

jqSlidingTab



Displays an area that can be expanded and collapsed by the user. The content is contained in a separate div and can be updated from the postback or initially set.

Properties:

Property	Type	Default	Description

tab	tabContent	Empty	tabContent class that describes the control such as the title and content HTML
initiallyOpen	Boolean	False	If True the control is created initially open and displays its content
callGetOnOpenClost	Boolean	False	The control does a post back when it is opened/closed, if this is True, the control will also do a Get which can return the content to load

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_page As String, ByVal p_submit_form As Boolean)`

Parameters	Type	Description
p_name	String	Identifier for the control
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected

Example:

```
Dim st As New clsJQuery.jqSlidingTab("myslide1_ID", "test", True)
st.initiallyOpen = True
st.tab.AddContent("the content")
st.tab.name = "myslide_name"
st.tab.tabName.Unselected = "Unselected Tab Title"
st.tab.tabName.Selected = "Selected Tab Title"
stb.Append(st.Build)
```

When the sliding tab is opened, a postback is performed, and the data will look as follows:

```
myslide1_ID=myslide_name_open
```

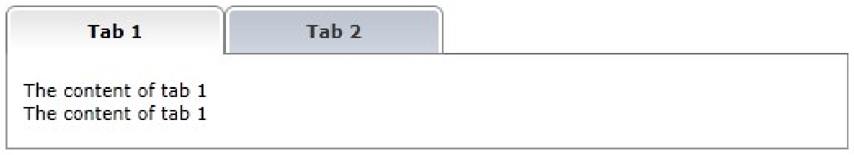
When the sliding tab is closed, it will not post back. If you need to know that it was closed, enable the "callGetOnOpenClose" property and the control will do a GET whenever the control is opened or closed.

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqTabs

jqTabs



The content of tab 1
The content of tab 1

Displays a number of tabs can contain HTML content. This control does not post back any data, it only holds content.

Properties:

Property	Type	Default	Description
tabs	Collections.Generic.List(Of tab)	Empty	Collection of tab objects, one for each tab

Creation:

`Public Sub New(p_name As String)`

Parameters	Type	Description
p_name	String	ID attribute to assign the controll

Example:

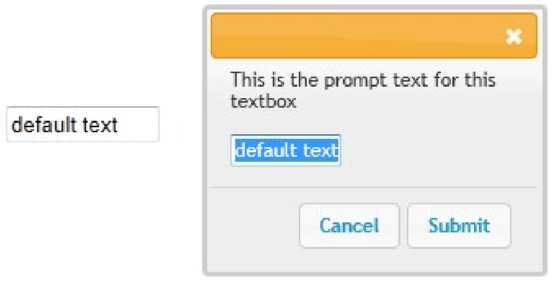
```
Dim jqtabs As New clsJQuery.jqTabs("tab1id")
Dim tab As New clsJQuery.Tab
tab.tabTitle = "Tab 1"
tab.tabContent = "The content of tab 1<br>The content of tab 1<br>"
jqtabs.tabs.Add(tab)

tab = New clsJQuery.Tab
tab.tabTitle = "Tab 2"
tab.tabContent = "The content of tab 2"
jqtabs.tabs.Add(tab)
stb.Append(jqtabs.Build)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTextBox
jqTimePicker
jqTimeSpanPicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqTextBox

jqTextBox



Displays a simple textbox for entering information. When a user clicks in the box, a small dialog appears where they can enter the information. When the user clicks submit on the dialog, a post back is performed and the data is posted back to your postBack function. The postback contains postdata in the format: name=value, where "name" is the name attribute. The name attribute is set with the ID as passed to New. If you are re-using this control you must reset the ID property.

Properties:

Property	Type	Default	Description
dialogWidth	Integer	0	If 0, the dialog width is automatically set based on the length of the default text. If > 0 then the dialog width is explicitly set
dialogCaption	String	"Edit Value"	The caption to display on the pop-up dialog
promptText	String	""	Label to display just above the textbox
modal	Boolean	False	True=dialog is modal, no other actions can be done on the page, False=non modal, other parts of the page can be accessed
editable	Boolean	False	If False, user cannot edit the text in the default textbox and when clicked, a dialog will display where the entry can be made. In this mode, a submit is performed from the dialog so the input text can be processed immediately. If True, the dialog is not displayed and the text may be entered into the textbox and is editable. The textbox is a standard HTML input box and the input text may be processed when the form is submitted.
formToPostID	String	""	If submitting the form, this can be set as the alternate form to post to

Creation:

Public Sub New(ByVal p_name As String, ByVal p_inputType As String, ByVal p_defaultText As String, ByVal p_page As String, ByVal p_size As Integer, ByVal p_submit_form As Boolean)

Parameters	Type	Description
p_name	String	name attribute to assign the controll
p_inputType	String	Input type, normally set to "text"
p_defaultText	String	Initial text that is displayed in the text box
p_size	Integer	Size of the initial text box in characters
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected

Example:

```
        Dim tb As New clsJQuery.jqTextBox("tb1", "text", "default text", "test", 10, True)
        tb.promptText = "This is the prompt text for this textbox"
        tb.toolTip = "This is the tooltip for this text box"
        tb.Name = "tb1"
        stb.Append(tb.Build)
```

When the user clicks on the submit button on the dialog, a postback occurs as follows (tb1=Name property, "12345" is the user entered text)

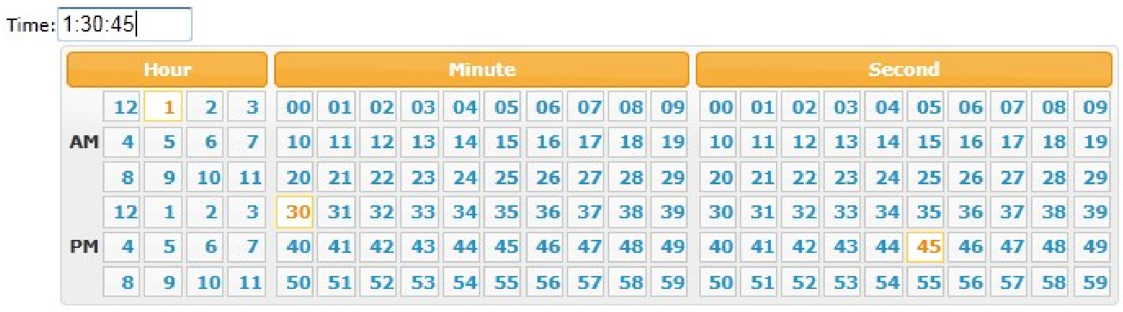
tb1=12345

If you want the ID to be posted, the ID can be set with:

tb.id=ID_TO_SET

See Also

jqTimePicker



Displays a dialog for picking a time.

Properties:

Property	Type	Default	Description
ampm	Boolean	False	If True, displays 12 hr format time, else 24 hr format
minutesSeconds	Boolean	False	If True, the dialog only displays minutes and seconds, otherwise the hour is also displayed
defaultValue	String	"12:00 PM"	The default time to display
editable	Boolean	False	If True, the user can edit the time in the default textbox
size	Integer	8	Size in characters of the default text box
showSeconds	Boolean	False	If True, the seconds are selectable
hourText	String	"Hour"	Section header for the hours
minuteText	String	"Minute"	Section header for the minutes
secondText	String	"Second"	Section header for the seconds
formToPostID	String	""	If submitting the form, this can be set as the alternate form to post to

Creation:

```
Public Sub New(ByVal p_id As String, ByVal p_label As String, ByVal p_page As String, ByVal p_submit_form As
Boolean)
```

Parameters	Type	Description
p_id	String	ID attribute to assign the controll

p_label	String	Label to display next to the default text box
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected

Example:

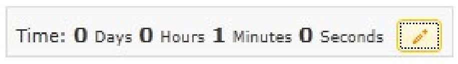
```
        Dim tp As New clsJQuery.jqTimePicker("mytm", "Time:", "test", True)
        tp.toolTip = "This is the tooltip for the Time Picker"
        tp.ampm = True
        tp.showSeconds = True
        tp.minutesSeconds = False
        tp.defaultValue = "1:30:45"
        stb.Append(tp.Build)
```

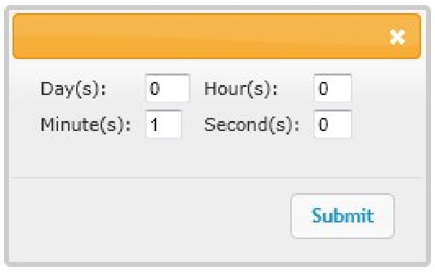
See Also

- jqButton
- jqBlockUI
- jqCheckBox
- jqColorPicker
- jqContainer
- jqDatePicker
- jqDropList
- jqDynSpinner
- jqFileUploader
- jqListBox
- jqListBoxEx
- jqLocalFileSelector
- jqMultiSelect
- jqOverlay
- jqProgressBar
- jqRadioButton
- jqScrollingRegion
- jqSelector
- jqSlider
- jqSlidingTab
- jqTabs
- jqTextBox
- jqTimeSpanPicker
- jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqTimeSpanPicker

jqTimeSpanPicker


Time: 0 Days 0 Hours 1 Minutes 0 Seconds


Day(s): 0 Hour(s): 0
Minute(s): 1 Second(s): 0

Displays a dialog for picking a span of time.

Properties:

Property	Type	Default	Description
defaultTimeSpan	TimeSpan	0,0,1,0,0	The default timespan to display

showSeconds	Boolean	True	If True, the seconds are selectable
showDays	Boolean	True	If True, the days are selectable
modal	Boolean	False	If True, no other action can be performed on the page
submitEnabled	Boolean	True	If False, no post back is performed when the submit button is pressed
formToPostID	String	""	If submitting the form, this can be set as the alternate form to post to

Creation:

`Public Sub New(ByVal p_name As String, ByVal p_label As String, ByVal p_page As String, ByVal p_submit_form As String)`

Parameters	Type	Description
p_name	String	Name attribute to assign the controll
p_label	String	Label to display next to the default text box
p_page	String	Page the control is on
p_submit_form	Boolean	If True, the entire form that the control is on is posted after a date is selected

Example:

```
        Dim ts As New clsJQuery.jqTimeSpanPicker("tm1", "Time:", "test", True)
        ts.toolTip = "This is the tooltip for the timespan picker"
        ts.showSeconds = True
        stb.Append(ts.Build)
```

When the user clicks the submit button, a postback occurs with the new timespan like:

```
tm1=2.2:1:30
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqToolTip

Home > Webpages > PageBuilder (clsPageBuilder) > JQuery Controls (clsJquery) > jqToolTip

jqToolTip

Displays an icon that can be hovered over to display a tooltip.

Note that all controls already have a tooltip property that can be set with text that will display when the mouse hovered over the control. This control can be used in cases where you might want to explicitly display helpful information.

This control does not post back any data.

Properties:

Property	Type	Default	Description
tipClass	String	"tooltip"	To change the format of the tooltip, enter your own class name

Creation:

```
Public Sub New(ByVal _title As String)
```

Parameters	Type	Description
p_title	String	The text of the tooltip to display when the mouse is hovered over the icon

Example:

```
stb.Append(New clsJQuery.jqToolTip("This is the tool tip!").build)
```

See Also

jqButton
jqBlockUI
jqCheckBox
jqColorPicker
jqContainer
jqDatePicker
jqDropList
jqDynSpinner
jqFileUploader
jqListBox
jqListBoxEx
jqLocalFileSelector
jqMultiSelect
jqOverlay
jqProgressBar
jqRadioButton
jqScrollingRegion
jqSelector
jqSlider
jqSlidingTab
jqTabs
jqTextBox
jqTimePicker
jqTimeSpanPicker

Home > Webpages > PageBuilder (clsPageBuilder) > AddAjaxHandlerPost

AddAjaxHandlerPost

```
Public Function AddAjaxHandlerPost(ByVal postData As String, ByVal page As String) As String
```

Adds a handler that is called when the page timer expires. The page timer is added with the property RefreshIntervalMilliSeconds. The parameter is data that is to be posted back to the given page. Normally this some combination of name/value pairs. For example:

```
stb.Append(Me.AddAjaxHandlerPost("action=updatetime", "pluginpage"))
```

This will create a function that is called that will post the "action=updatetime" to the page named "pluginpage".

Parameters:
postData: The data that is posted when the timer expires. If the data contains a "$" it is assume that the data is actually script and it is not enclosed in single quotes.

page: The page to post back to.

See Also

Building a Page
reset
AddHeader
AddBody
AddFooter
suppressDefaultFooter
BuildPage
DivStart
DivEnd
FormStart
FormEnd
divToUpdate
pageCommands
JQuery Controls (clsJquery)

Home > Webpages > Page Performance

Page Performance

By default, a log of javascript code is included on a web page, this is due to the vast amount of jquery controls that we support. Jquery is included as "includes" at the top of every page. For most pages, not all of that javasctipt code is required. To speed up page loading, you can specifiy which jquery controls you want to include. At the start of your GetPage function you can add:

UsesJqAll=False

This removes all the custom controls from the page, but still includes the standard jquery so controls like droplist, textbox, button, and slidingtab still can be used. To include other jquery controls, set any of the following to True:

```
UsesJqTimePicker

UsesJqDatePicker

UsesjqTimeSpanPicker

UsesjqLocalFileSelector

UsesJqColorPicker

UsesjqBlockUI

UsesjqMultiSelect

UsesjqFileUploader

UsesjqCheckBox

UsesJqTabs

UsesjqDynSpinner

UsesjqScrollingRegion

UsesjqListBox
```

If you do not add anything to your page, UsesJqAll is set to True and everything is included.

See Also

GetPagePlugin
PageBuilder (clsPageBuilder)
PostBackProc
GenPage
PagePut
RegisterConfigLink
RegisterHelpLink
RegisterLink
UnRegisterHelpLinks
ASP.NET

PostBackProc

Public Function PostBackProc(ByVal pageName As String, ByVal data As String, ByVal user As String, ByVal userRights As Integer) As String Implements HomeSeerAPI.IPlugInAPI.PostBackProc

When a user clicks on any controls on one of your web pages, this function is then called with the post data. You can then parse the data and process as needed.

Parameters:
pageName: the name of the page as registered with hs.RegisterLink or hs.RegisterConfigLink
data: the post data
user: name of logged in user
userRights: rights of logged in user

Returns: any serialized data that needs to be passed back to the web page, generated by the clsPageBuilder class

Here is a sample web page that uses the clsPageBuilder class to build a complete web page. This class would be created when the plugin starts and when PostBackProc was called in the plugin it would call the postBackProc in this class.

```
Imports System.Text
Imports System.Web
Imports Scheduler

Public Class WebPage
    Inherits clsPageBuilder

    Public Sub New(ByVal pagename As String)
        MyBase.New(pagename)
    End Sub

    Public Overrides Function postBackProc(page As String, data As String, user As String, userRights As
Integer) As String
        Dim parts As Collections.Specialized.NameValueCollection
        parts = HttpUtility.ParseQueryString(data)

        ' handle postbacks here
        ' update DIV'S with:
        'Me.divToUpdate.Add(DIV_ID, HTML_FOR_DIV)
        ' refresh a page (this page or other page):
        'Me.pageCommands.Add("newpage", url)
        ' open a dialog
        'Me.pageCommands.Add("opendialog", "dyndialog")
        If parts("id") = "b1" Then
            Me.divToUpdate.Add("current_time", "This div was just updated with this")
        End If
        If parts("action") = "updatetime" Then
            ' ajax timer has expired and posted back to us, update the time
            Me.divToUpdate.Add("current_time", DateTime.Now.ToString)
            If DateTime.Now.Second = 0 Then
                ' disable the spinner on the div named updatediv
                Me.pageCommands.Add("stopspinner", "updatediv")
                Me.divToUpdate.Add("updatediv", "job complete")
            ElseIf DateTime.Now.Second = 30 Then
                ' start the spinner on the div named updatediv
                Me.pageCommands.Add("startspinner", "updatediv")
                Me.divToUpdate.Add("updatediv", "working...")
            End If
        End If

        Return MyBase.postBackProc(page, data, user, userRights)
    End Function


    ' build and return the actual page
    Public Function GetPagePlugin(ByVal pageName As String, ByVal user As String, ByVal userRights As Integer,
ByVal queryString As String) As String
        Dim stb As New StringBuilder
        Try
            Me.reset()

            ' handle any queries like mode=something
            Dim parts As Collections.Specialized.NameValueCollection = Nothing
            If (queryString <> "") Then
                parts = HttpUtility.ParseQueryString(queryString)
            End If

            ' add any custom menu items

            ' add any special header items
            'page.AddHeader(header_string)

            ' add the normal title
            Me.AddHeader(hs.GetPageHeader("Sample Plugin", "", "", True, False))

            ' add the standard menus
            'stb.Append(Me.AddMenuBar())

            stb.Append(clsPageBuilder.DivStart("pluginpage", ""))

            ' a message area for error messages from jquery ajax postback (optional, only needed if using AJAX
calls to get data)
            stb.Append(clsPageBuilder.DivStart("errormessage", "class='errormessage'"))
            stb.Append(clsPageBuilder.DivEnd)

            ' specific page starts here

            stb.Append("<div id='current_time'>" & DateTime.Now.ToString & "</div>" & vbCrLf)

            Dim b As New clsJQuery.jqButton("b1", "Button", IFACE_NAME, False)
            stb.Append(b.Build)
            stb.Append(clsPageBuilder.DivEnd)

            ' add the body html to the page
            Me.AddBody(stb.ToString)

            Me.AddFooter(hs.GetPageFooter)
            Me.suppressDefaultFooter = True

            ' return the full page
            Return Me.BuildPage()
        Catch ex As Exception
            'WriteMon("Error", "Building page: " & ex.Message)
            Return "error"
        End Try
    End Function

End Class
```

See Also

GetPagePlugin
PageBuilder (clsPageBuilder)
Page Performance
GenPage
PagePut
RegisterConfigLink
RegisterHelpLink
RegisterLink
UnRegisterHelpLinks
ASP.NET

Home > Webpages > GenPage

GenPage

```
Public Function GenPage(ByVal link As String) As String Implements HomeSeerAPI.IPlugInAPI.GenPage
```

This function is available for the ease of converting older HS2 plugins, however, it is desirable to use the new clsPageBuilder class for all new development.

This function is called by HomeSeer from the form or class object that a web page was registered with using RegisterConfigLink. You must have a GenPage procedure per web page that you register with HomeSeer. This page is called when the user requests the web page with an HTTP Get command, which is the default operation when the browser requests a page.

The lnk parameter passed into this procedure is the full URL that the user typed. Thus, if your web page is **\acme_rockets**, but the user added parameters such as **\acme_rockets?Name1=Value1&Name2=Value2**, then that full link (URL) will be passed to GenPage.

Within GenPage, you can process the name/value pairs in the lnk parameter passed to it, or you can use the procedure GetFormData provided in the sample plug-in. GetFormData breaks the URL into name/value pairs, which is a custom type array **(tPair)** also provided in the sample plug-in, and sets the variable **lPairs** to the number of name/value pairs provided.

GenPage returns a string of HTML, which is the heart of your plug-in web page. HomeSeer will generate the page headers/footers, so your plug-in only has to supply the body of the web page.

The sample plug-in has many utility functions to help with the generation of HTML code - see the HTMLPublic.vb module in the sample plug-in for more information.

When creating your output, you must have a field named **ref_page** included in the HTML, with the value set to the web page link. This field allows HomeSeer to determine which plug-in is providing this page for purposes of handling PUT requests, which is discussed more in the PagePut document. Here is a sample of adding this reference to the output using the AddHidden utility function from the sample plug-in that adds hidden fields to the web page:

If you are writing your web pages in ASP.NET, your GenPage function will simply re-direct to the ASP.NET page. Assuming your web page was named *widget.aspx* and it exists in the directory *widget* under the HomeSeer HTML directory, GenPage would look like:

```
     Public Function GenPage(ByVal lnk As String) As String

        Dim sb As New StringBuilder

        Dim data As New StringBuilder

        data.Append("<HEAD>" & vbCrLf)

        data.Append("<meta http-equiv=""refresh"" content=""0;url=widget/widget.aspx"">" &
vbCrLf)

        data.Append("</HEAD>" & vbCrLf)

        sb.Append("HTTP/1.0 200 OK" & vbCrLf)

        sb.Append("Server: HomeSeer" & vbCrLf)

        sb.Append("Expires: Sun, 22 Mar 1993 16:18:35 GMT" & vbCrLf)

        sb.Append("Content-Type: text/html" & vbCrLf)

        sb.Append("Accept-Ranges: bytes" & vbCrLf)

        sb.Append("Content-Length: " & data.Length & vbCrLf & vbCrLf)

        sb.Append(data.ToString)

        Return sb.ToString

        'GenPage = BuildPage()

     End Function
```

See Also

- GetPagePlugin
- PageBuilder (clsPageBuilder)
- Page Performance
- PostBackProc
- PagePut
- RegisterConfigLink
- RegisterHelpLink
- RegisterLink
- UnRegisterHelpLinks
- ASP.NET

Home > Webpages > PagePut

PagePut

```
Public Function PagePut(ByVal data As String) As String Implements HomeSeerAPI.IPlugInAPI.PagePut
```

When your plug-in web page has form elements on it, and the form is submitted, this procedure is called to handle the HTTP "Put" request. There must be one PagePut procedure in each plug-in object or class that is registered as a web page in HomeSeer.

PagePut is responsible for handling the fields that were in the form, and then it returns what you want HomeSeer to display after the Put operation is complete. Most of the time, the return from PagePut is another call to GenPage to generate a new, refreshed web page.

PagePut is passed a **data** parameter which contains the form data. This information is URL Encoded, which means that binary or unprintable characters are converted to an ASCII representation of their HEX values. The sample plug-in uses GetFormData, to process this data, which besides putting the data into individual Name/Value pairs in an array, handles the conversion by calling UrlDecode as well. (UrlDecode is included in the sample plug-in as well.)

After processing the data, you may set variables, save INI settings, or do other things as directed by the user in the web page, and that may change how the returned page is drawn by GenPage. The sample plug-in has two web pages built into it - one with the main HSPI class of the plug-in, and another in the WebLink1 module. The page in the HSPI module shows a button on the web page toggling a variable that controls whether HomeSeer generates the web page header/footer or if the plug-in is responsible for that functionality. As you toggle the RAW format, notice that the title of the web page changes from the one it was registered with, to the one added by the manual web page creation mode in GenPage.

See Also

- GetPagePlugin
- PageBuilder (clsPageBuilder)
- Page Performance
- PostBackProc
- GenPage
- RegisterConfigLink
- RegisterHelpLink
- RegisterLink
- UnRegisterHelpLinks
- ASP.NET

Home > Webpages > RegisterConfigLink

RegisterConfigLink

```
Public Sub RegisterConfigLink(ByVal cbo As webPageDesc) Implements IAppCallbackAPI.RegisterConfigLink
```

This call is functionally identicle to RegisterLink, except that the link does NOT display on the HomeSeer links menu. It is only available from the config button on the HomeSeer interfaces page.

See Also

GetPagePlugin
PageBuilder (clsPageBuilder)
Page Performance
PostBackProc
GenPage
PagePut
RegisterHelpLink
RegisterLink
UnRegisterHelpLinks
ASP.NET

RegisterHelpLink

```
Public Sub RegisterHelpLink(ByVal cbo As webPageDesc) Implements IHSApplication.RegisterHelpLink
```

This call is functionally identicle to RegisterLink, except that the link does NOT display on the HomeSeer links menu. It is only available from the Help menu.

See Also

GetPagePlugin
PageBuilder (clsPageBuilder)
Page Performance
PostBackProc
GenPage
PagePut
RegisterConfigLink
RegisterLink
UnRegisterHelpLinks
ASP.NET

RegisterLink

```
Public Sub RegisterLink(ByVal cbo As webPageDesc) Implements IAppCallbackAPI.RegisterLink
```

This call registers a web page from your plug-in with HomeSeer, which causes not only the link to work, but for HomeSeer to add the page to the link menu displayed at the top of the HomeSeer web pages.

Parameters: cbo web page description class, the class has the following properties:

plugInName: the name of the plugin
plugInInstance: the instance of the plugin (if SupportsMultipleInstances returns TRUE)
link: the text of the link that will be used to access the page

linktext: the display text of the link (can be the same as the link)
page_title: a title to be used for the page (displays in the html but is not visible)

Returns: Nothing

Example:

```
' register a configuration link that will appear on the interfaces page
 Dim wpd As New webPageDesc
 wpd.link = IFACE_NAME
 wpd.linktext = "Sample Plugin Config"
 wpd.page_title = "Sample Plugin Config"
 wpd.plugInName = IFACE_NAME
 callback.RegisterConfigLink(wpd)

 ' register a normal page to appear in the HomeSeer menu
 wpd = New webPageDesc
 wpd.link = IFACE_NAME
 wpd.linktext = "Sample Plugin Page"
 wpd.page_title = "Sample Plugin Page"
 wpd.plugInName = IFACE_NAME
 callback.RegisterLink(wpd)
```

See Also

GetPagePlugin
PageBuilder (clsPageBuilder)
Page Performance
PostBackProc
GenPage
PagePut
RegisterConfigLink
RegisterHelpLink
UnRegisterHelpLinks
ASP.NET

Home > Webpages > UnRegisterHelpLinks

UnRegisterHelpLinks

Public Sub UnRegisterHelpLinks(ByVal plugin_name As String, plugin_instance As String) Implements IHSApplication.UnRegisterHelpLinks

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

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

See Also

GetPagePlugin
PageBuilder (clsPageBuilder)
Page Performance
PostBackProc
GenPage
PagePut
RegisterConfigLink
RegisterHelpLink
RegisterLink
ASP.NET

ASP.NET

ASP.NET is supported for web pages, however, it is recommended that you use the jquery controls that HomeSeer provides along with the clsPageBuilder class. To use ASP.NET use the GenPage and PutPage functions.

To access a plugin from an ASPX page, use the PluginAccess class. This class is a wrapper for all available plugin API's. In your Page_Load function initialize a new PluginAccess object and pass it a reference to the HomeSeer object, the plugin name and instance, and the name of the plugin you want to access. Check the "Connected" property to make sure you are connected to the plugin.

The test.aspx page that is included with HomeSeer in the html folder has some sample code that accesses a plugin.

Here is a sample Page_Load function:

```
Sub Page_Load(Sender As Object, E As EventArgs)
    ' get a reference to the HomeSeer API
    hs = Context.Items("Content")

    ' get a reference to the Z-Wave plugin
    plugin = New HomeSeerAPI.PluginAccess(hs, "Z-Wave", "")

    If plugin.Connected Then
        Label2.Text = "Connected to Z-Wave plugin, name=" & plugin.Name
    End If

    label1.text = "HomeSeer Ver: " & hs.version

end sub
```

See Also

GetPagePlugin
PageBuilder (clsPageBuilder)
Page Performance
PostBackProc
GenPage
PagePut
RegisterConfigLink
RegisterHelpLink
RegisterLink
UnRegisterHelpLinks

Home > Speak Proxy

Speak Proxy

Articles in this section

SpeakIn
SpeakProxy

See Also

Document Revisions
Getting Started
Controlling with JSON
Plugin Initialization
Base Plugin API
Devices
Callbacks
Triggers
Actions
Webpages
Script ASP
Technology APIs
Updater
Appendices

Home > Speak Proxy > SpeakIn

SpeakIn

SpeakIn

SpeakIn(device As Short, text As String, wait As Boolean, host As String)

If your plug-in is registered as a Speak proxy plug-in, then when HomeSeer is asked to speak something, it will pass the speak information to your plug-in using this procedure. When your plug-in is ready to do the actual speaking, it should call SpeakProxy, and pass the information that it got from this procedure to SpeakProxy. It may be necessary or a feature of your plug-in to modify the text being spoken or the host/instance list provided in the host parameter - this is acceptable.

The parameters are as follows:

device - This is the device that is to be used for the speaking. In older versions of HomeSeer, this value was used to indicate the sound card to use, and if it was over 100, then it indicated that it was speaking for HomeSeer Phone (device - 100 = phone line), or the WAV audio device to use. Although this is still used for HomeSeer Phone, speaks for HomeSeer phone are never proxied and so values >= 100 should never been seen in the device parameter. Pass the device parameter unchanged to SpeakProxy.

text - This is the text to be spoken, or if it is a WAV file to be played, then the characters ":\" will be found starting at position 2 of the string as playing a WAV file with the speak command in HomeSeer REQUIRES a fully qualified path and filename of the WAV file to play.

wait - This parameter tells HomeSeer whether to continue processing commands immediately or to wait until the speak command is finished - pass this parameter unchanged to SpeakProxy.

host - This is a list of host:instances to speak or play the WAV file on. An empty string or a single asterisk (*) indicates all connected speaker clients on all hosts. Normally this parameter is passed to SpeakProxy unchanged.

See Also

SpeakProxy

SpeakProxy

SpeakProxy

SpeakProxy(device As Short, text As String,

Optional wait As Boolean = False,

Optional host As String = "")

This procedure is used to cause HomeSeer to speak something when a speak proxy is registered and active. Since speak commands when a speak proxy plug-in is registered are trapped and passed to the SpeakIn procedure of the speak proxy plug-in, this command is used when the speak proxy plug-in is ready to do the real speaking.

The parameters are identical to those described in SpeakIn - please refer to that document for parameter information.

See Also

SpeakIn

Script ASP

Articles in this section

THIS SECTION IS NOT COMPLETE

See Also

Script_and_ASP_Encryption

Script and ASP Encryption

There are two types of encryption that you may use for your scripts, and one type for your ASP pages. The first type uses encoding that Microsoft developed and was made available for use with their older, original script technologies - VBScript and JavaScript. This type of encoding is by no means secure - it is not recommended that you use it; in fact, it is so old, that "good luck" in trying to find the encoding program at the Microsoft web sites. It should also be noted that scripts encoded using Microsoft's encoding cannot be called with parameters.

HomeSeer's encryption is stronger and uses a key known to HomeSeer so that the scripts can be opened and interpreted by HomeSeer (naturally). You can encrypt your script or ASP page by running it through the HomeSeer Script Encoder utility, which will read the source file and produce a destination (encrypted) file as a result. When you use the utility, you will use a different extension on the encrypted file so that HomeSeer knows how to process it. Here is a list of the file types and their file extensions.

File Type	Normal (Unencrypted) Extension	Encrypted Extension
Visual Basic Script	**.TXT, .VBS**	**.VBH (HomeSeer Encryption)** **.VBE (Microsoft Encoding)**
.NET Visual Basic Script	**.VB**	**.VBEN (HomeSeer Encryption)**
Java Script	**.JS**	**.JSH (HomeSeer Encryption)** **.JSE (Microsoft Encoding)**
Active Server Pages	**.ASP**	**.ASH (HomeSeer Encryption)**

.NET Active Server Pages	**.ASPX**	**Unsupported! ASPX pages are directly interpreted by .NET and therefore cannot be encrypted.**

Since ASPX technology does not allow the encryption of these pages, they should instead use .NET "code behind" technology which produces a .NET software (DLL) module which accompanies the ASPX page. As much of the application's code that can be put in the code behind page, the more secure it will be, and this is also where you will need to put the license check call discussed in the licensing topic of this section. Note that .NET DLLs are not encrypted and can be reverse engineered, and so therefore your best protection in this case is to dotfuscate the code making it more difficult to understand when it is reviewed by a .NET code interpreter.

The HomeSeer script encoder utility is included with this SDK. To use the utility, follow these simple instructions:

1. Run the utility
2. Select your source file - you may do this using one of two ways:
    1. Enter the full path and filename in the "File Selection" text box on the form, then press the "Source" button at the bottom of the form.
    2. Select the drive where your source file is located using the drop down list of drives in your system, then choose the path to the file location by clicking on the folders in the folder browser, and then click on the filename of your file in the file list window on the right-hand side of the window. After you click the filename, the full path and filename should appear in the "File Selection" box, then click the "Source" button at the bottom of the form.
3. Select your destination file - you may do this using one of two ways:
    1. Enter the full path and filename in the "File Selection" text box on the form, then press the "Destination" button at the bottom of the form.
    2. Select the drive where your destination file is to be located using the drop down list of drives in your system, then choose the path to the file location by clicking on the folders in the folder browser, and then click on the filename of your source file in the file list window on the right-hand side of the window if it is located in this directory, and then change the file extension in the "File Selection" box - alternatively you can click on any file in the destination path so that the full path and filename appears in the "File Selection" box, then replace the filename with the one you wish to use, making sure to use the proper file extension from the above table. The full path and filename should appear in the "File Selection" box, then click the "Destination" button at the bottom of the form.
        - If your destination file already exists, you will be warned at the top of the form that your destination file will be overwritten if you complete the process.
4. Press the "GO" button at the bottom of the form - the encrypted file will be produced, and the utility will then prompt you to do another file, and if you do not desire to do another file, the utility will end.

Click here to access the HomeSeer Script Encoder Utility

Right-Click and use "Save As" to store a copy of the utility on your PC in a folder of your choosing.

See Also

Script_and_ASP_Encryption_and_Licensing
Script_and_ASP_Licensing

Home > Script ASP > Script_and_ASP_Encryption_and_Licensing

Script_and_ASP_Encryption_and_Licensing

Script and ASP Encryption and Licensing

The only element needed from HomeSeer for a script or ASP based application is licensing. Licensing only works if the user is unable to thwart licensing or copy the software, and this is only done with encryption. Thus, to provide a licensed script or ASP based software system, you will need to encrypt it and add some lines of script code to provide a license mechanism. Review the following topics to learn more about this.

- Script and ASP Encryption
- Script and ASP Licensing

See Also

Script_and_ASP_Encryption
Script_and_ASP_Licensing

Home > Script ASP > Script_and_ASP_Licensing

Script_and_ASP_Licensing

Script and ASP/ASPX Licensing

To use the script license verification function in your script or ASP, you will have had to previously register your script with HomeSeer and file a License ID with us for your script. When you file a License ID with us, we add it to our software that generates licenses and thus the ability for the script to be licensed through the HomeSeer On-Line store is possible. After we generate a LF (license file) and send it to you, it will be necessary for you to include this file with your updater package for your script or ASP/ASPX system. You should copy this file to the user's CONFIG directory, making sure NOT to copy the file if it already exists.

The first time a licensed script is run and your script makes the function call described later in this section, HS2 will automatically grant the user a 30 day trial period for that script. The return code of the function call below will alert you as to whether the script is running in a trial mode, and thus you may opt to NOT continue the script if you do not desire to have a trial period for your script or system.

When you register your script with HomeSeer, there is some information that needs to be provided to HST. You must provide a License ID string and a Product ID string. The LicenseID string is a short, unique string for your system. For example, if you wrote a call handling system for HomeSeer Phone, you might provide a License ID of CALL-PROC. The Product ID is a longer, more descriptive string which is used in informational messages to the user (if they are not suppressed in the function call) so the user better understands which system it is that generated the message in the log. For this example again, we might provide a Product ID string of Full featured call handling system for HomeSeer Phone by Charlie Axehandle. This string is also what we will use by default at the HomeSeer Online Store for the description of the product.

In your script, you will need to make the following function call if you wish to invoke license checking for it. Failure to do so will still allow the encrypted script to run it will provide protection for your intellectual property (coding) but not the actual output of the script or ASP.

Function ValidateScriptLicense(LicenseID As String, ProductID As String, Optional bDisplay As Boolean=True) As Integer

The parameters are as follows:

- LicenseID = The short license string for your script as registered with HomeSeer.
- ProductID = The longer text description of the script or ASP system.
- BDisplay = A Boolean value indicating whether you want HomeSeer to report licensing information in the HomeSeer log. If set to True, the default, then HomeSeer will write notices that indicate that the script system is unlicensed, running under a trial copy, etc. It will not display anything for a fully licensed script. If you prefer to handle license notifications to the end-user, pass a False value for this argument.

Return Value

The integer return value indicates the registration for the script or ASP/ASPX. Here is a breakdown of the return values:

- -1 = Trial period has expired, or user has never registered this script.
- 0-99 = Trial period. The number returned indicates the number of days remaining. (Typically a maximum of 30)
- 100 = Trial period. This type of trial is provided by a HomeSeer issued trial registration code. The period of time remaining is not returned, but if bDisplay is True the expiration date is shown in the log file.
- 999 = Fully registered.

Generally, you would check to see if a user is authorized to run your script with the following code:

```
Dim i_ret

i_ret = hs.ValidateScriptLicense(c_license_id, "My script system is great.", True)
If i_ret = -1 Then
    'User's license does not exist or has expired. HS will report this to the log.
    'Do not run script.
    Exit Sub    ' Or Exit Function
End If
```

In this case, HomeSeer will check to see how many days of the trial period are left, and if they have expired, will log a note in the log notifying the user of this fact. You would place code, such as that above, in an initialization routine and/or at the top of a critical bit of code that would be executed with some regularity.

If you do not want to have HomeSeer write expiration messages in the log for you, you have complete control over how to handle the trial period as with the following example:

Here is an example of using ValidateScriptLicense to control access to your script (call this function early in your script) This example denies a trial period:

```
Dim iReturn
Dim LicenseID

LicenseID = "MY-SYSTEM"

iReturn = hs.ValidateScriptLicense(LicenseID, "My script system is great.", True)

If iReturn <> 999 Then
    hs.WriteLog LicenseID, "Sorry - unregistered.  Contact HomeSeer to purchase a license."
    Exit Sub    ' Or exit function - get out of the system.
End If

'  This is where your script´s "really neat stuff" happens.
```

Remember, .NET compiled libraries are not encrypted - they can be read with a tool that interprets the .NET runtime commands. When placing license check procedures in your code behind page of your ASPX project, be sure to dotfuscate the final DLL file.

See Also

Script_and_ASP_Encryption
Script_and_ASP_Encryption_and_Licensing

Technology APIs

Articles in this section

This sections descripes the programming of technology specific APIs for plugins.

See Also

Document Revisions
Getting Started
Controlling with JSON
Plugin Initialization
Base Plugin API
Devices
Callbacks
Triggers
Actions
Webpages
Speak Proxy
Script ASP
Updater
Appendices

Thermostat API

See the Device Type section of the scripting reference for details on how to create devices that represent thermostats, music systems, etc.

The Thermostat API has been removed and is now replaced with the Device Type as a means of enumerating through devices to find devices representing thermostat functionality. All of the functionality of a thermostat should exist with a collection of "child" devices associated to a single root "parent" device.   Using the device type information, once a device is found which indicates it is a thermostat API device, the rest of the devices associated with the same parent are considered together as "a single thermostat".

When the device type information indicates a thermostat API device, the specific device type will indicate which thermostat functionality the device belongs to, and only in the case of a "setpoint" device, the device sub-type information is used to indicate which setpoint the device represents in the situation where the thermostat supports multiple discrete setpoints for the different operating modes.

See Also

Media API

Media API

The Media API in HS3 is vastly different from previous versions.   The Media API now uses devices for most functionality, and uses a slimmed down API for working with the library, the playlists, and for starting media playing.

Enumerating a media API from a HomeSeer system can be done two different ways, but both will be needed for complete control of a media system. The first method for discovering a media system is to enumerate through the devices looking for a device type that indicates the Media API. Once a device is found that uses the Media API, the device's Interface and InterfaceInstance properties may be used to get the plug-in name and instance that controls the device, and that will grant you access to the parts of the API implemented only in the plug-in. Note: The child device should have the Interface and InterfaceInstance property populated properly, but in case it is not, please refer to the root "parent" device to get this information.

The other method is the reverse of the first one wherein the plug-ins can be enumerated, and any plug-in returning a Capabilities value with bit 6 set (Hex=20), indicates that it is a Media API plug-in. The plug-in can then be queried to determine if it supports multiple instances and if so, what the instance names are, and that information can be used to find the matching devices for each instance.

The enums player_state_values, repeat_modes, and player_selections are used by devices, and are defined in the API for consistency between media plug-ins. As with these enums and any others, contact HomeSeer Technologies to request a change.

The remainder of this section documents the API calls made into a Media API supporting Plug-In.

See Also

Thermostat API

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins

Media API Procedures for Plug-Ins

The old Music API from previous versions of HomeSeer have evolved considerably in this version of HomeSeer HS3. Most APIs have been replaced by device types which hold information on how to interact with plug-ins. The library portion of the Music API could not exist and be worked with using the device metaphor, so it is the only API that still exists - in part - as an API in HS3.

The Music API was renamed to the Media API, and only procedures dealing with the media library exist in the API. All other player controls, status, triggers and other capabilities that used to be in the API are now to be expressed in devices using the value/status pairs to indicate status and provide control for the players. See the Device Class documentation in the scripting reference for information about the Device Type class and the special device types that exist for media devices.

The remainder of this section deals with the media library procedures, which includes information on specific tracks/pieces of media, how it is organized and categorized, and playlists containing several pieces of media.

The purpose of these API commands is not to modify the media library directly - that is the function of the media plug-in - but rather to provide a way for user interfaces to work with the library to display or select media to be played by the players.

See Also

Device Enum Values

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Playing Media

Playing Media

These procedures are used to load the currently playing playlist with media entries and starting playback of those entries.

See Also

Working with the Library
Getting Library Entries
Getting Library Types
Working with Saved Playlists
Working with the Current Playlist

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Playing Media > Play

Play

This procedure is used to start playback of a specific media entry from the current playlist, or from the library if the current playlist is empty. If a list of media to be played has already been chosen (the currently playing list is populated) then the control device Play command should be used if a specific start track is not indicated.

```
Sub Play(ByVal Key As Lib_Entry_Key)
```

See Also

PlayPlaylist / PlayPlaylistAt
PlayGenre / PlayGenreAt
PlayMatch

PlayPlaylist / PlayPlaylistAt

These commands are used to start the playback of a specific pre-set playlist of media. The Playlist_Entry structure is provided to indicate the playlist name and library type where the playlist can be found, and PlayPlaylistAt is used when the playback of the playlist is desired to start on a specific entry in the list.

When these commands are called, the currently playing playlist should be cleared if it contains any entries.

```
Sub PlayPlaylist(ByVal Playlist As Playlist_Entry )

Sub PlayPlaylistAt(ByVal Playlist As Playlist_Entry , ByVal Start_Key As Lib_Entry_Key )
```

See Also

Play
PlayGenre / PlayGenreAt
PlayMatch

PlayGenre / PlayGenreAt

These commands start playback of all of the media fitting the genre name, library type, and entry type specified. PlayGenreAt can be used when playback of the selected tracks should start with a known entry.

When these commands are called, the currently playing playlist should be cleared if it contains any entries.

```
Sub PlayGenre(ByVal GenreName As String, _
              ByVal Lib_Type As UInt16, _
              ByVal EntryType As eLib_Media_Type)

Sub PlayGenreAt(ByVal GenreName As String, _
                ByVal Lib_Type As UInt16, _
                ByVal EntryType As eLib_Media_Type, _
                ByVal Start_Key As Lib_Entry_Key)
```

See Also

Play
PlayPlaylist / PlayPlaylistAt
PlayMatch

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Playing Media > PlayMatch

PlayMatch

This procedure is used to start playback of the media which matches the selections provided in the Play_Match_Info structure. If the Playlist item in the structure is populated with the name of a valid Playlist in the library, then the remaining items are used to filter which items in the Playlist will be played. If the Playlist item is not populated or contains an invalid playlist name, then the remaining items are used to select media from the library to be played. A null (empty) value or an asterisk (*) in any field is used to indicate "all", meaning that field is not used as a filter in the selection. Similarly, a library type (L0_Lib_Type) of 0 indicates all media libraries, and a media type (Lib_Media_Type) of "Any_All" (99) indicates no filtering of the media types.

```
Sub PlayMatch(ByVal MatchInfo As Play_Match_Info )
```

See Also

Play
PlayPlaylist / PlayPlaylistAt
PlayGenre / PlayGenreAt

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Playing Media > PlayMatch > Play_Match_Info Structure

Play_Match_Info Structure

This structure is used by the PlayMatch API command:

```
Public Structure Play_Match_Info

      Public Lib_Media_Type As eLib_Media_Type  ' Plays media matching the media type provided (Any_All=99)
      Public L0_Lib_Type As UInt16              ' Plays media from the library matching the Library Type
      (0=Any/All)
      Public L1_Genre As String                 ' Plays media with the matching level 1 selector (Commonly
      used as the Genre).
      Public L2_Artist As String                ' Plays media with the matching level 2 selector (Commonly
      used as the Artist).
      Public L3_Album As String                 ' Plays media with the matching level 3 selector (Commonly
      used as the Album).
      Public L4_Title As String                 ' Plays media with the matching level 4 selector (Commonly
      used as the Title).
      Public Playlist As String                 ' If populated with a value and a corresponding playlist
      exists, the found
                                                  playlist is played and is filtered by the values in the
      other selectors.

End Structure
```

See Also

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Working with the Library

Working with the Library

Body of text here

See Also

Playing Media
Getting Library Entries
Getting Library Types
Working with Saved Playlists
Working with the Current Playlist

LibLoading

Before making calls to get information about the media library, get this property to determine if the library is currently being updated from the library source. If this property returns True, the library content is being updated, and your plug-in should wait until it returns False, or indicate in the return to the end user that the library content may be incomplete because the library is currently being updated.

```
ReadOnly Property LibLoading As Boolean
```

See Also

Library Count

Library Count

These functions return a count of the number of entries in the media library total (Count), that match an entry type (CountbyEntryType), that match a library type (CountbyLibType), or that match both entry type and library type (CountbyType).

There are other functions that retrieve library entries using a subset (starting point and count) if the entire library - use these functions to know the limit of the number of entries to be retrieved with those functions.

```
Function LibGetLibraryCount() As Integer
Function LibGetLibraryCountbyEntryType(ByVal EntryType As eLib_Media_Type) As Integer
Function LibGetLibraryCountbyLibType(ByVal Lib_Type As UInt16) As Integer
Function LibGetLibraryCountbyType(ByVal Lib_Type As UInt16, ByVal EntryType As eLib_Media_Type) As Integer
```

See Also

LibLoading

eLib_Media_Type

This enum is referenced by the Lib_Entry structure to define the library entry type.

```
Public Enum eLib_Media_Type
    Unknown_Error = 0
    Music = 1
    Video = 2
    Music_Stream = 3
    Video_Stream = 4
    Photo = 5
    Radio = 6
    Radio_Stream = 7
    Playlist = 8
    Other_09 = 9
    Other_10 = 10
    Other_11 = 11
    Other_12 = 12
    Other_13 = 13
    Other_14 = 14
    Other_15 = 15
    Any_All = 99
End Enum
```

Note: With some functions, providing an entry type of 0 (zero) indicates a "don't care" situation to match all entry types. Therefore, the use of 0 as a returned library entry type is invalid.

See Also

Getting Library Entries

The functions in this procedure return entries from the media library, which may then be used to set the current playlist.

See Also

Playing Media
Working with the Library
Getting Library Types
Working with Saved Playlists
Working with the Current Playlist

Lib_Entry Structure

This structure is used to define the library entry return from the library calls of the Music API.

```
Public Structure Lib_Entry

    Public Lib_Entry_Type As eLib_Media_Type
    Public Lib_Type As UInt16
    Public Key As Lib_Entry_Key
    Public Title As String
    Public Artist As String
    Public Album As String
    Public Genre As String
    Public LengthSeconds As UInt32
    Public Cover_path As String
    Public Cover_Back_path As String
    Public Year As UInt16
    Public Rating As Byte
    Public PlayedCount As UInteger
    Public Kind As String

End Structure
```

See Also

LibGet Tracks, Albums, Artists, Genres
LibGetLibrary
LibGetLibrarybyLibType
LibGetLibraryRange
LibGetEntry

eLib_Media_Type

This enum is referenced by the Lib_Entry structure to define the library entry type.

```
Public Enum eLib_Media_Type
    Unknown_Error = 0
    Music = 1
    Video = 2
    Music_Stream = 3
    Video_Stream = 4
    Photo = 5
    Radio = 6
    Radio_Stream = 7
    Playlist = 8
    Other_09 = 9
    Other_10 = 10
    Other_11 = 11
    Other_12 = 12
    Other_13 = 13
    Other_14 = 14
    Other_15 = 15
    Any_All = 99
End Enum
```

Note: With some functions, providing an entry type of 0 (zero) indicates a "don't care" situation to match all entry types. Therefore, the use of 0 as a returned library entry type is invalid.

See Also

Lib_Entry_Key Structure

Lib_Entry_Key Structure

This structure is used to provide the key and the description of the key type used, when referencing entries in a media library.

```
Public Structure Lib_Entry_Key

    Public iKey As Integer  ' Used for numerical key based systems.

    Public sKey As String   ' Used for non-numerical key based systems.
    Public WhichKey As eKey_Type  ' Used to describe which of the above two key systems is valid for this entry.

    Public Library As UInt16           ' Ties to Lib_Type structure, Lib_Type_Num member.
End Structure
```

See Also

eLib_Media_Type

eKey_Type

This Enum defines which of the two key system types are used by a key entry, the Enum values are as follows:

```
Public Enum eKey_Type
    eUnspecified_Error = 0
    eNumber = 1
    eString = 2
    eEither = 3
End Enum
```

When creating a new Lib_Entry_Key, care should be taken to specifically initialize the WhichKey (eKey_Type) so that it does not remain at its default value of 0 (Error).

See Also

LibGet Tracks, Albums, Artists, Genres

Each of these functions select a subset of the media library and return information. Only LibGetTracks returns an array of lilbrary entry keys which may be used to set a current playlist for playback. LibGetAlbums, Artists, and Genres are used to retrieve the names of the possible choices for Album names, Artist names, and Genres from the library.

```
Function LibGetTracks(ByVal artist As String, _
                      ByVal album As String, _
                      ByVal genre As String, _
                      ByVal Lib_Type As UInt16) As Lib_Entry_Key()

Function LibGetAlbums(ByVal artist As String, _
                      ByVal genre As String, _
                      ByVal Lib_Type As UInt16) As String()

Function LibGetArtists(ByVal album As String, _
                       ByVal genre As String, _
                       ByVal Lib_Type As UInt16) As String()

Function LibGetGenres(ByVal Lib_Type As UInt16) As String()
```

Note: If the Lib_Type provided is the reserved value 0, matching entries from all libraries will be returned (0 = Any/All Libraries)

See Also

Lib_Entry Structure
LibGetLibrary
LibGetLibrarybyLibType
LibGetLibraryRange
LibGetEntry

LibGetLibrary

LibGetLibrary is used to get the contents of the entire media library - all entry types, all library types. Use this function conservatively and with caution, as the operation may take a long time and require a large amount of memory to perform.

```
Function LibGetLibrary() As Lib_Entry()
```

See Also

Lib_Entry Structure
LibGet Tracks, Albums, Artists, Genres
LibGetLibrarybyLibType
LibGetLibraryRange
LibGetEntry

LibGetLibrarybyLibType

LibGetLibrarybyLibType is used to get the contents of the media library matching the given library type - all entry types. Use this function conservatively and with caution, as the operation may take a long time and require a large amount of memory to perform.

```
Function LibGetLibrarybyLibType(ByVal Lib_Type As UInt16) As Lib_Entry()
```

See Also

Lib_Entry Structure
LibGet Tracks, Albums, Artists, Genres
LibGetLibrary
LibGetLibraryRange
LibGetEntry

LibGetLibraryRange

The LibGetLibraryRange functions are used to retrieve a subset of the media library. It can be called to retrieve a subset of the entire library, a subset of the library which matches an entry type, a subset of the library that matches a library type, or a subset of the library that matches both entry type and library type.

```
    Function LibGetLibraryRange(ByVal Start As Integer, ByVal Count As Integer) As Lib_Entry()

    Function LibGetLibraryRangebyEntryType(ByVal Start As Integer, ByVal Count As Integer, _
                                           ByVal EntryType As eLib_Media_Type) As Lib_Entry()
    Function LibGetLibraryRangebyLibType(ByVal Start As Integer, ByVal Count As Integer, _
                                         ByVal LibType As UInt16) As Lib_Entry()

    Function LibGetLibraryRangebyType(ByVal Start As Integer, ByVal Count As Integer, _
                                      ByVal LibType As UInt16, _
                                      ByVal EntryType As eLib_Media_Type) As Lib_Entry()
```

Note: Unlike LibGetTracks, LibGetAlbums, LibGetArtists, and LibGetGenres, these functions can NOT use a LibType of 0 to indicate 'any type' - the library type must be specified and a value of 0 should result in an empty or null list return.

See Also

- Lib_Entry Structure
- LibGet Tracks, Albums, Artists, Genres
- LibGetLibrary
- LibGetLibrarybyLibType
- LibGetEntry

LibGetEntry

LibGetEntry is used to get one specific library entry, using its key.

```
Function LibGetEntry(ByVal Key As Lib_Entry_Key) As Lib_Entry
```

See Also

- Lib_Entry Structure
- LibGet Tracks, Albums, Artists, Genres
- LibGetLibrary
- LibGetLibrarybyLibType
- LibGetLibraryRange

Lib_Entry_Key Structure

This structure is used to provide the key and the description of the key type used, when referencing entries in a media library.

```
Public Structure Lib_Entry_Key

    Public iKey As Integer  ' Used for numerical key based systems.

    Public sKey As String   ' Used for non-numerical key based systems.

    Public WhichKey As eKey_Type  ' Used to describe which of the above two key systems is valid for this entry.

    Public Library As UInt16          ' Ties to Lib_Type structure, Lib_Type_Num member.

End Structure
```

See Also

eKey_Type

This Enum defines which of the two key system types are used by a key entry, the Enum values are as follows:

```
Public Enum eKey_Type
    eUnspecified_Error = 0
    eNumber = 1
    eString = 2
    eEither = 3
End Enum
```

When creating a new Lib_Entry_Key, care should be taken to specifically initialize the WhichKey (eKey_Type) so that it does not remain at its default value of 0 (Error).

See Also

Getting Library Types

The media library may be made up of a combination of different library types - e.g. from a computer based media library such as iTunes or Windows Media Player, or from an on-line library resource such as Pandora, Rhapsody, or Spotify, or perhaps even and on-line streaming service such as Netflix. The Library Types are defined by the media plug-in, and this function is used to retrieve a list of those library types, and their associated library type values.

```
    Function LibGetLibraryTypes() As Lib_Type()
```

See Also

Playing Media
Working with the Library
Getting Library Entries
Working with Saved Playlists
Working with the Current Playlist

Lib_Type Structure

This structure maps a friendly name for a library type to its unique Type number.

```
Public Structure Lib_Type
    Public Lib_Type_Num As UInt16    ' Do NOT use the value 0 - initialize upon creation with a value > 0.
    Public Lib_Name As String
End Structure
```

See Also

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Working with Saved Playlists

Working with Saved Playlists

Body of text here

See Also

Playing Media
Working with the Library
Getting Library Entries
Getting Library Types
Working with the Current Playlist

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Working with Saved Playlists > Playlist_Entry Structure

Playlist_Entry Structure

This structure is used in the playlist related API calls:

```
Public Structure Playlist_Entry
    Public Lib_Type As UInt16          ' Value 0 is reserved
    Public Playlist_Name As String
    Public Playlist_Key As UInt16      ' Value 0 indicates the key is not used.
    Public Length As UInt32            ' Number of entries in the playlist.
End Structure
```

Note: The value 0 for Lib_Type is reserved for use as an indicator of 'any/all' library types.

See Also

Lib_Entry_Key Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylists
LibGetPlaylistTracks

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Working with Saved Playlists > Lib_Entry_Key Structure

Lib_Entry_Key Structure

This structure is used to provide the key and the description of the key type used, when referencing entries in a media library.

```
Public Structure Lib_Entry_Key

    Public iKey As Integer  ' Used for numerical key based systems.

    Public sKey As String   ' Used for non-numerical key based systems.

    Public WhichKey As eKey_Type  ' Used to describe which of the above two key systems is valid for this entry.
    Public Library As UInt16            ' Ties to Lib_Type structure, Lib_Type_Num member.
End Structure
```

See Also

Playlist_Entry Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylists
LibGetPlaylistTracks

eKey_Type

This Enum defines which of the two key system types are used by a key entry, the Enum values are as follows:

```
Public Enum eKey_Type
    eUnspecified_Error = 0
    eNumber = 1
    eString = 2
    eEither = 3
End Enum
```

When creating a new Lib_Entry_Key, care should be taken to specifically initialize the WhichKey (eKey_Type) so that it does not remain at its default value of 0 (Error).

See Also

Playlist_Add

Use this function with a Playlist_Entry structure to add a new playlist to the system. If the return is False, then the playlist name and library type specified in the Playlist_Entry parameter are not unique in the system (see Playlist_Delete).

```
    Function Playlist_Add(ByVal Playlist As Playlist_Entry) As Boolean
```

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Delete
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylists
LibGetPlaylistTracks

Playlist_Delete

Use this function to remove a playlist from the system. This does not remove the media library entries that are in this playlist from the media library, it only removes the playlist from the system.

```
Function Playlist_Delete(ByVal Playlist As Playlist_Entry) As Boolean
```

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Add
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylists
LibGetPlaylistTracks

Playlist_Add_Track

Use this function to add the selected media entry (Key) to the indicated Playlist (Playlist_Entry).

```
Function Playlist_Add_Track(ByVal Playlist As Playlist_Entry, ByVal TrackKey As Lib_Entry_Key) As Boolean
```

Note: The Lib_Entry that the Lib_Entry_Key is associated with includes a library type, and the Playlist (Playlist_Entry) also includes a library type. Generally entries from one library may not be intermixed within another library type, so it is up to the plug-in author to return a failure condition when the library types do not match.

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Tracks
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylists
LibGetPlaylistTracks

Playlist_Add_Tracks

Use this function to add the selected media entries (TrackKeys) to the indicated Playlist (Playlist_Entry).

```
Function Playlist_Add_Tracks(ByVal Playlist As Playlist_Entry, ByVal TrackKeys As Lib_Entry_Key()) As Boolean
```

Note: The Lib_Entry that the Lib_Entry_Key is associated with includes a library type, and the Playlist (Playlist_Entry) also includes a library type. Generally entries from one library may not be intermixed within another library type, so it is up to the plug-in author to return a failure condition when the library types do not match. It is also up to the plug-in author to determine how to handle the array of Lib_Entry_Key and duplicate values, whether to not add them or to add duplicates so that the resulting playlist has multiple copies of some tracks.

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Track
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylists
LibGetPlaylistTracks

Playlist_Delete_Track

This function removes the indicated (Lib_Entry_Key) library entry from the indicated (Playlist_Entry) playlist.

```
Function Playlist_Delete_Track(ByVal Playlist As Playlist_Entry, ByVal TrackKey As Lib_Entry_Key) As Boolean
```

Note: It is up to the plug-in author to determine if the removal of a library entry that does not exist in the playlist is an error condition (Not found, return False) or non-error condition (already removed, return True).

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Tracks
LibGetPlaylists
LibGetPlaylistTracks

Playlist_Delete_Tracks

This function removes the indicated (Lib_Entry_Key) library entries from the indicated (Playlist_Entry) playlist.

```
Function Playlist_Delete_Tracks(ByVal Playlist As Playlist_Entry, ByVal TrackKeys As Lib_Entry_Key()) As Boolean
```

Note: It is up to the plug-in author to determine if the removal of a library entry that does not exist in the playlist is an error condition (Not found, return False) or non-error condition (already removed, return True). It is also up to the plug-in author to determine whether the failure of a single entry from the array constitutes a False return or a True return is some of the entries were removed.

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Track
LibGetPlaylists
LibGetPlaylistTracks

LibGetPlaylists

This command retrieves all of the playlists from the system.

```
Function LibGetPlaylists(Optional ByVal Lib_Type As UInt16 = 0) As Playlist_Entry()
```

If the Lib_Type is provided, only the playlists belonging to the indicated library type should be returned. Lib_Type of 0 means return all playlists regardless of the library type.

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylistTracks

LibGetPlaylistTracks

Once a playlist has been selected, its playlist entry structure can be used to retrieve all of the members of the playlist.

```
Function LibGetPlaylistTracks(ByVal Playlist As Playlist_Entry) As Lib_Entry()
```

See Also

Playlist_Entry Structure
Lib_Entry_Key Structure
Playlist_Add
Playlist_Delete
Playlist_Add_Track
Playlist_Add_Tracks
Playlist_Delete_Track
Playlist_Delete_Tracks
LibGetPlaylists

Working with the Current Playlist

The "Current Playlist" can exist virtually within the plug-in, or electronically/in-memory in the media player device or program. In either situation, it is treated as a non-saved playlist meaning that it is not guaranteed to be static between restarts of the plug-in or the media player device/program. The current playlist represents the media currently being served by the media player. Procedures which call for a specific piece of media or a playlist to be played are not directly referenced as the current playlist - the selected media or playlist is to be copied to the current playlist before playback begins.

The PlayPlaylist and PlayPlaylistAt commands should always clear the current playlist before their entries are copied to the current playlist.

See Also

Playing Media
Working with the Library
Getting Library Entries
Getting Library Types
Working with Saved Playlists

Lib_Entry_Key Structure

This structure is used to provide the key and the description of the key type used, when referencing entries in a media library.

```
Public Structure Lib_Entry_Key
    Public iKey As Integer  ' Used for numerical key based systems.
    Public sKey As String   ' Used for non-numerical key based systems.
    Public WhichKey As eKey_Type  ' Used to describe which of the above two key systems is valid for this entry.
    Public Library As UInt16          ' Ties to Lib_Type structure, Lib_Type_Num member.
End Structure
```

See Also

Getting Current PlayList Information
Modifying the Current Playlist

eKey_Type

This Enum defines which of the two key system types are used by a key entry, the Enum values are as follows:

```
Public Enum eKey_Type
    eUnspecified_Error = 0
    eNumber = 1
    eString = 2
    eEither = 3
End Enum
```

When creating a new Lib_Entry_Key, care should be taken to specifically initialize the WhichKey (eKey_Type) so that it does not remain at its default value of 0 (Error).

See Also

Getting Current PlayList Information

See Also

Lib_Entry_Key Structure
Modifying the Current Playlist

CurrentlyPlaying

This function returns a Lib_Entry_Key structure for the Lib_Entry currently playing in the media player. An empty/null return indicates that there is no media currently playing.

```
Function CurrentlyPlaying() As Lib_Entry_Key
```

See Also

CurrentPlayListCount
CurrentPlayList / CurrentPlayListRange

CurrentPlayListCount

This function returns the number of entries in the 'current playlist'

```
Function CurrentPlayListCount() As Integer
```

See Also

CurrentlyPlaying
CurrentPlayList / CurrentPlayListRange

CurrentPlayList / CurrentPlayListRange

These functions return Lib_Entry_Key structure arrays containing the contents of the player's "current playlist". Use CurrentPlayListCount to determine the upper bounds of the playlist count before using CurrentPlayListRange.

```
Function CurrentPlayList() As Lib_Entry_Key()
Function CurrentPlayListRange(ByVal Start As Integer, ByVal Count As Integer) As Lib_Entry_Key()
```

See Also

CurrentlyPlaying
CurrentPlayListCount

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Working with the Current Playlist > Modifying the Current Playlist

Modifying the Current Playlist

The commands listed herein modify the "current playlist".

See Also

Lib_Entry_Key Structure
Getting Current PlayList Information

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Working with the Current Playlist > Modifying the Current Playlist > CurrentPlayListSet

CurrentPlayListSet

CurrentPlayListSet is used to set the current playlist, and it is provided with an array of Lib_Entry_Key values containing the keys to the tracks that should be set. This command always clears the current playlist before setting the list of tracks to the current playlist.

```
Function CurrentPlayListSet(ByVal Tracks As Lib_Entry_Key()) As Boolean
```

See Also

CurrentPlayListAdd
CurrentPlayListClear

Home > Technology APIs > Media API > Media API Procedures for Plug-Ins > Working with the Current Playlist > Modifying the Current Playlist > CurrentPlayListAdd

CurrentPlayListAdd

CurrentPlayListAdd works much the same as CurrentPlayListSet, except that the tracks (array of Lib_Entry_Key) are to be added to whatever tracks may already be in the "current playlist".

```
Function CurrentPlayListAdd(ByVal Tracks As Lib_Entry()) As Boolean
```

See Also

CurrentPlayListSet
CurrentPlayListClear

CurrentPlayListClear

This command will clear the "current playlist". It is up to the plug-in author's discretion to determine whether playback of the currently playing media should be stopped immediately or if playback will end when the currently playing media has ended.

```
Sub CurrentPlayListClear()
```

See Also

CurrentPlayListSet
CurrentPlayListAdd

Device Enum Values

See Also

Media API Procedures for Plug-Ins

player_state_values

```
Public Enum player_state_values

      Idle = 0   ' Player is idle or at a default state undetermined, no media is loaded to play
      playing = 1   ' Player is actively playing
      stopped = 2   ' Player is stopped, but there may be media to play
      paused = 3   ' Player is paused on the current media being played
      forwarding = 4   ' Player is scanning the current media forward/ahead
      rewinding = 5   ' Player is scanning the current media backward/reverse
      MediaEnding = 6  ' The current media is ending
      MediaStarting = 7  ' A new media entry is starting to play
      Waiting = 8   ' Player is waiting on network congestion, media buffering, for user input, etc.
      Playlist_Changed = 9  ' The playlist of media to be played has changed.
      Library_Updating = 10  ' The media library is being updated - may only indicate this status at the start
      of the update.

End Enum
```

See Also

repeat_modes
shuffle_modes
player_selections

repeat_modes

```
Public Enum repeat_modes

      repeat_off = 1
      repeat_one = 2
      repeat_all = 3

End Enum
```

See Also

player_state_values
shuffle_modes
player_selections

Home > Technology APIs > Media API > Device Enum Values > shuffle_modes

shuffle_modes

```
Public Enum shuffle_modes

      not_shuffled = 1
      shuffled = 2

End Enum
```

See Also

player_state_values
repeat_modes
player_selections

Home > Technology APIs > Media API > Device Enum Values > player_selections

player_selections

```
** Note: This enum comes from HS2 and is deprecated in HS3 **

Public Enum player_selections

      Track = 1
      Album = 2
      Artist = 3
      Album_Track = 4
      Artist_Track = 5
      Artist_Album = 6
      Artist_Album_Track = 7
      Playlist = 8
      Playlist_Track = 9
      Genre = 10
      All_Tracks = 11
      URL = 12
      Unknown_Nothing = 0

End Enum
```

See Also

player_state_values
repeat_modes
shuffle_modes

Updater

All plugins must use the updater ZIP package format as described in this section. MSI installations are no longer supported. The ZIP format will ensure that all plugins may be installed under Windows as well as Linux. MSI packages are not supported in windows.

The package format is mainly unchanged from HS2, however, the topics in this section have not been reveiwed yet and may not be complete.

THIS TOPICS IN THIS SECTION MAY NOT BE COMPLETE

Articles in this section

See Also

Updater Packaging Introduction

Most plug-ins and scripts are delivered to the end user through the HomeSeer Updater. The updater is a web page that contains most of the UI elements for the user to select packages to be downloaded and installed, and HomeSeer does the rest with procedures built into the HomeSeer scheduler. The updater is accessed from the Plugins->Manage menu.

If you are creating a plug-in, then the product has to have a unique product ID string (for plug-ins, this is what is returned from the *Name* function), so that a product can be created for it in our licensing system. You must also provide information to HomeSeer and agree to the terms set forth by HST to use the HST store, licensing system, and for maintenance of the licenses and license information. Please visit **this web page** for all of the latest information and requirements.

Instructions for creating a package and testing it are in the sections under this topic. When you are sending a package to HomeSeer to be placed in the updater, please follow these general rules and best practice suggestions:

- Send your package description HTML document that is used for the updater package description only when there is a change made to it. Do not include it with your package's updater zip file - attach it to the email "as is" or if you like, zip it up in a separate zip file from the updater package.
- Send your package icon graphic for the updater only when there is a change made to it, and just like the description document, do not put it in the updater install package zip file.
- Name the installation package zip file using the package version number. If your plug-in was called 'Foo' for example, then your package zip file might be named **Foo_1-2-0-3.zip**

You have the option of hosting the update package on your own server. If you do so, you only need to send us the updater line that contains the URL to your server. You can then post updates yourself and you do not have to notify us. See the Updater Line Format topic for information on how to host your own update.

Package updates should be sent to **Updater@HomeSeer.com.** Attach your package zip to the email but rename the extension to .zzz otherwise the mail system may reject it.

HomeSeer Plugin Submission process checklist

First time submission:

1. Send an email to updater@homeseer.com and include the following:
    1. The name of your plugin (this is the name that is returned from the Name property in your plugin
    2. Description of your plugin, see other plugin descriptions for a format
    3. Your full name
    4. If you are charging for this plugin, the amount you are going to charge
    5. If you are charging for this plugin, the paypal address where payments are sent
    6. An icon to be used in the updater. The icon format is 32x32 pixels.
    7. The zip package that will be posted to the updater. You get a sample zip package by looking in your HS3 install folder updates3/zips. Updates that you have installed will be located there.
    8. Include an updater line in the email for your package. See the Update Line Format section for details.
    9. Before submitting your package, test it with the procedure in the Testing Your Package topic.
    10. If you are selling your plugin, follow the instructions here for getting your plugin in our store.

Subsequent submissions:

1. Send an email to updater@homeseer.com with an updated zip package, or an updated updater line if you are hosting the update.

See Also

Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
Copy
DELALL
DELFILES
File_Locations_-_Directory_Standards
FONT
INI
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Updater_Structure
Update Line Format

Home > Updater > Installation Script Index

Installation Script Index

Note: Each entry in the install.txt file is a command there are no commands exceeding a single line. All parameters are separated with commas (,). When a parameter is a text string, commas that you wish to have in the output can be indicated by using the characters %2C and nothing else. For example, the function to display a message box has a parameter that is the message you want to display. If you want the message to read Go drink a cold Coca Cola, now my friend. then you would write that parameter in this file as: Go drink a cold Coca Cola%2C now my friend.

(default)	Copy a file or delete a single file
[MSGBOX]	Display a message to the user with a message box
[CHECKVERSION]	Require a specific version of HomeSeer (or higher)
[LOCALCOPY]	Copy a local file from one place to another under the HomeSeer directory structure
[LOCALCOPYNONFATAL]	Instruct the updater to not fail on error copying local files
[ASK] or [ASKN]	Ask the user a question and skip over part of the install.txt file if the answer is NO
[ASKY]	Ask the user a question and skip over part of the install.txt file if the answer is YES
[ISINSTALLER]	Run a 3rd party InstallShield installer program
[INSTALLER]	Run a 3rd party installer program with optional parameters
[INIADD]	Add to an existing INI key entry
[INIADDPARM]	Add a parameter to an existing INI key entry (comma separated)
[INI]	Modify (replace) or add a new key to an INI file
[DELFILES]	Delete all of the files in a directory
[DELALL]	Delete a directory and all of the files and subdirectories in the directory
[UNZIP]	Unzip a zip archive to a directory, do not overwrite existing files
[UNZIPOVER]	Unzip a zip archive to a directory, overwrite existing files
[FONT]	Copy and install a font into the system

See Also

ASKY

ASKY

Ask the user a question and skip over part of the install.txt file if the answer is YES.

Parameter 1: Destination label to continue processing commands at with a YES answer.

Parameter 2: [ASKY]

Parameter 3: Question text to be displayed for the user.

The label specified in Parameter 1 is used on a line by itself preceeded by a colon (:) to indicate the continuation point.

Example:

```
,[MSGBOX],This is the beginning of the install.txt file.
FIRSTTEST,[ASKY],Answer YES if you want me to go to FIRSTTEST.
,[MSGBOX],You answered NO to the FIRSTTEST question.
SECONDtest,[ASKY],Answer this one YES to go to SECONDTEST.
:FIRSTTEST
,[MSGBOX],I have just passed the FIRSTTEST mark.
:SEcOnDTesT
,[MSGBOX],I have just passed the SECONDTEST mark.
```

(Destination labels are not case sensitive.)

Back to the Installation Script Index

See Also

ASK_or_ASKN

ASK or ASKN

Ask the user a question and skip over part of the install.txt file if the answer is NO.

Parameter 1: Destination label to continue processing commands at with a NO answer.

Parameter 2: [ASK] or [ASKN]

Parameter 3: Question text to be displayed for the user.

The label specified in Parameter 1 is used on a line by itself preceeded by a colon (:) to indicate the continuation point.

Example:

```
,[MSGBOX],This is the beginning of the install.txt file.
FIRSTTEST,[ASK],Answer NO if you want me to go to FIRSTTEST.
,[MSGBOX],You answered YES to the FIRSTTEST question.
SECONDtest,[ASK],Answer this one NO to go to SECONDTEST.
:FIRSTTEST
,[MSGBOX],I have just passed the FIRSTTEST mark.
:SEcOnDTesT
,[MSGBOX],I have just passed the SECONDTEST mark.
```

(Destination labels are not case sensitive.)

Back to the Installation Script Index

See Also

CHECKVERSION

CHECKVERSION

Require a specific version of HomeSeer (or higher)

Parameter 1: (not used)

Parameter 2: [CHECKVERSION]

Parameter 3: 4 position dotted decimal version to check for.

This feature should be the first line in your install.txt file so that none of the remaining installation commands take place.

Example:

To require version 1.6.182 or higher of HomeSeer for your package:

```
xxxx,[CHECKVERSION],1.6.0.182
```

Back to the Installation Script Index

See Also

Contents_of_an_Updater_Submission

Contents of an Updater Submission

A submission to the updater consists of a package zip file with the installation script and the files to be installed (more details soon). When it is the first submission or when they change, you would also include a package description HTML document, and a package icon file. Here is more information on each of those components:

Description File

The description file (document) is an HTML document that provides information regarding the package for the end user. See some available updates for a sample of this file. This file will be installed either on your server or ours. The updater.txt file includes the URL where this file it.

Icon

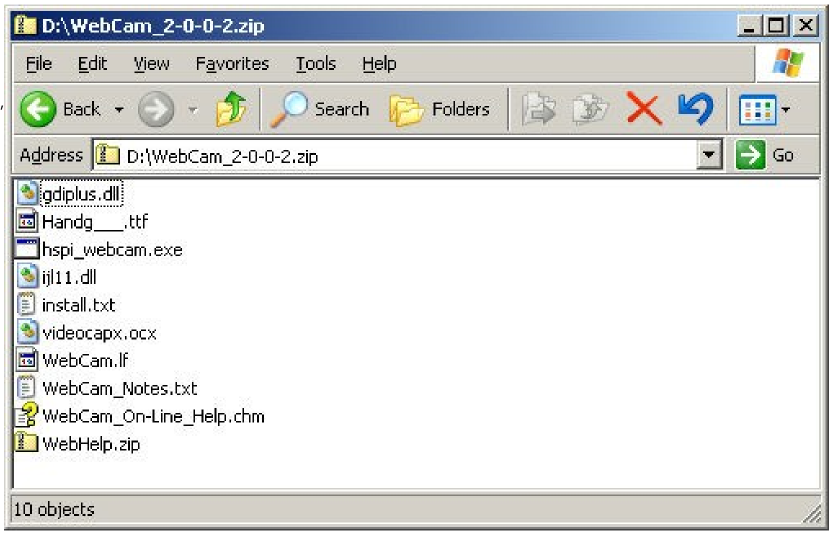
The icon can be any web graphic format, suggested formats include GIF, JPG, or PNG. The size of the icon is CURRENTLY not limited by the updater, but please refrain from using icons greater than 32x32px in size. GIF files with transparency characteristics usually have the best appearance. The URL to this icon is in the updater.txt file.

Installation Package

The installation package is a zip file, and as already mentioned it should be named in such a way as to include the name of the product and the version number. An installation package zip file contains all of the files to be installed on the remote system and must also have an installation script file called **INSTALL.TXT**. The installation script file commands are detailed in the Installation Script section of this document. If your package is a licensed plug-in, please remember to also copy your application's license file (.LF extension) to the HomeSeer config folder as a part of the package installation. The copy should be done such that an existing LF file is not overwritten, which for the application 'Foo' would look like this script command:

```
Foo.lf,.\Config,16
```

When many files need to be copied to a location, you may use a zip file within the package zip file. There are installation script commands to unzip a zip file with or without overwriting existing files. Here is an example of the contents of an installation package for the WebCam plug-in by HomeSeer - as you can see, it contains several files, a package license (LF) file, a font file to be installed, a Windows HTML help file, as well as a WebHelp HTML help document system enclosed in the WebHelp.zip file.


D:\WebCam_2-0-0-2.zip
File Edit View Favorites Tools Help
Back Search Folders
Address D:\WebCam_2-0-0-2.zip Go

```
gdiplus.dll
Handg___.ttf
hspi_webcam.exe
ijl11.dll
install.txt
videocapx.ocx
WebCam.lf
WebCam_Notes.txt
WebCam_On-Line_Help.chm
WebHelp.zip
```

10 objects

The installation script INSTALL.TXT in the above package looks like this (blank lines inserted for readability):

```
,[CHECKVERSION],1.6.0.139

gdiplus.dll,[WINSYS],17

WebCam_Notes.txt,.\Docs,0

WebCam_On-Line_Help.chm,.\Help,0

WebHelp.zip,[UNZIPOVER],.\html\WebCam

VideoCapX.ocx,.,5

ijl11.dll,.,16

hspi_webcam.exe,.,6

WebCam.lf,.\Config,16

Handg___.ttf,[FONT],HandelGothic
```

The full details of the package script are in the script section, but I will cover what is in this example in brief so you have an understanding of what is going on in a somewhat typical installation:

- Line 1 uses [CHECKVERSION] to make sure that the user's HomeSeer system is on at least version 1.6.0.139 - the installation does not continue if this condition is not met.
- Line 2 copies the file gdiplus.dll to the user's Windows System directory, and the updater determines that directory that is at runtime. The number at the end of the line indicates copy options explained in more detail in the script section, but in this case it indicates that the file should only be copied if it does not already exist, and it is a DLL or OCX file that should be registered with the operating system after it is installed.
- Line 3 copies the file WebCam_Notes.txt to the user's Docs directory under the root HomeSeer directory.
- Line 4 copies the HTML help file to the user's Help directory under the root HomeSeer directory.
- Line 5 unzips the file WebHelp.zip to the user's HTML\WebCam directory, which it will create if necessary, and it unzips the contents of the zip file with permission to overwrite any existing files that may already be in the HTML\WebCam directory.
- Line 6 copies a control (videocapx.ocx) used by the plug-in to the user's root HomeSeer directory, and the copy options indicate that the file should be unregistered before the new file is copied, and then the new file should be registered with the operating system.
- Line 7 copies a DLL file to the root HomeSeer folder, but the copy options indicate that it should only be copied if it does not already exist.
- Line 8 copies the main plug-in program to the root HomeSeer folder, and the copy options indicate that it is an EXE that should be unregistered if it already exists before the new version is copied over, and then the new version should be registered with the operating system.
- Line 9 copies the plug-in license file (webcam.lf) to the user's CONFIG folder under the HomeSeer root directory, providing that the file does not already exist there.
- Line 10 installs the font file, Handg___.ttf to the user's system with the font name of HandelGothic.

Again, please review the options in the Installation Script section for more information on the capabilities of the updater install script.

See Also

- Updater Packaging Introduction
- Installation Script Index
- ASKY
- ASK_or_ASKN
- CHECKVERSION
- Copy
- DELALL
- DELFILES
- File_Locations_-_Directory_Standards
- FONT
- INI
- INIADD
- INIADDPARM
- INSTALLER
- ISINSTALLER
- LOCALCOPY
- LOCALCOPYNONFATAL
- Testing Your Package
- UNZIP
- UNZIPOVER
- Updater_Structure
- Update Line Format

Copy

Copy a file or delete a single file

Parameter 1: The filename of the file to install such as weather.txt.

Parameter 2: The destination directory for the file. This is always relative to the root HomeSeer directory. To install a plug-in file to the HomeSeer directory, use .. To install a file into the scripts directory, use .\Scripts The special destination path of **[WINSYS]** can also be used to copy the file to the user's SYSTEM directory.

Parameter 3: This value is a numeric bit field that describes options that should be applied to the install. The value entered here is the decimal representation of the bits. The bits available are:

bit 0 = &h1 = 1	The file is an OCX or DLL file and needs to be registered using regsvr32.
bit 1 = &h2 = 2	The file is an EXE file and needs to be registered by calling the EXE file with the switch /regserver.
bit 2 = &h4 = 4	The file is an exe/ocx/dll file and should be unregistered. EXE files are unregistered with /unregserver (if it exists). DLL and OCX files are unregistered with "regsvr32.exe /u".
bit 3 = &h8 = 8	File is a script file and the function **hs_install** should be called after the package is installed. Make sure this file is the last one in the list if it depends on other files being installed first.
bit 4 = &h10 = 16	Do not install the file if the file already exists.
bit 5 = &h20 = 32	Delete the file if it exists.

Set this parameter to 0 if the file does not require any of the above options.

Bits may be "ORED" together to represent multiple options. For example, to register an exe file, but do not copy the file if the target already exists, enter: 18 (bit 1 and bit 4 = 12hex or 18 dec)

Note: Please be familiar with the **File Locations - Directory Standards** document before creating your package an indeed before publishing your script, ASPX, or plug-in package.

Examples:

weather.txt,.\scripts,0

MainScript.vbs,.\scripts,8

NewInstall.vbs,.\scripts,24

hspi_Plugin.exe,.,6

hspi_Plugin.ocx,.,1

Back to the Installation Script Index

See Also

Updater Packaging Introduction
Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
DELALL
DELFILES
File_Locations_-_Directory_Standards
FONT
INI
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Updater_Structure
Update Line Format

Home > Updater > DELALL

DELALL

DELALL

Delete a directory and all of the files and subdirectories in the directory.

Parameter 1: (not used)

Parameter 2: [DELALL]

Parameter 3: The path and name of the directory you wish to have deleted with extreme prejudice. (Relative to the HS directory)

Examples:

xxx,[DELALL],.\scripts\includes\myapp

xxx,[DELALL],.\html\includes\myapp

Back to the Installation Script Index

See Also

Updater Packaging Introduction
Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
Copy
DELFILES
File_Locations_-_Directory_Standards
FONT
INI
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Updater_Structure
Update Line Format

Home > Updater > DELFILES

DELFILES

DELFILES

Delete all of the files in a directory.

Parameter 1: (not used)

Parameter 2: [DELFILES]

Parameter 3: The path and name of the directory you wish to have emptied. (Relative to the HS directory)

Examples:

```
xxx,[DELFILES],.\scripts\includes\myapp\dir1
xxx,[DELFILES],.\scripts\includes\myapp\dir2
xxx,[DELFILES],.\html\includes\myapp\dir1
xxx,[DELFILES],.\html\includes\myapp\dir2
```

Back to the Installation Script Index

See Also

File_Locations_-_Directory_Standards

File Locations - Directory Standards

It is important to maintain some structure in the use of directories when you install HomeSeer. Users want to know consistently where to go to find files. Here are the directory naming and usage standards to follow when creating your install.txt file. All paths in the following table are relative to the root HomeSeer directory.

(appname) refers to a form of your application's title suitable for use as a directory name. For example, a plug-in titled "Lutron HomeWorks" might use "HomeWorks" for (appname).

Type	Directory	Notes
Documentation	\Docs	Not help files, but readme files and other documents.
Configuration Settings	\Config or \Config\***(appname)*** if there are a lot of files for this location.	Do not use HomeSeer's SETTINGS.INI file.
Windows Electronic Help Files	\Help	e.g. CHM files. Should not be used with HomeSeer 2.x any longer.
HTML Help Files	\HTML\Help\***(appname)***	HTML generators such as Robohelp produce Webhelp packages with many files -- do not clutter up the shared help directory -- create your own for your application.
HTML	\HTML or \HTML\***(appname)***	
HTML Includes	\HTML\Includes\***(appname)***	
HTML Images	\HTML\Images\***(appname)***	
Scripts	\Scripts	

Script Includes or other script files.	\Scripts***(appname)*** or \Scripts\Includes\***(appname)***	
Data Files and/or Database Files	\DATA***(appname)***	The users just LOVE it when they can backup their data easily and know which directory to back up!
Binaries (executables)	\bin***(appname)***	
Temporary Files during install OR used by your application	\Temp or \Temp***(appname)***	Remember to clean up any mess (files) when you are done.

Note: There should not be any files in the HomeSeer root directory other than your main plug-in executable. If you feel your application needs an exemption from this standard, please let us know and we will be happy to explain why you are wrong. ;-)

See Also

Updater Packaging Introduction
Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
Copy
DELALL
DELFILES
FONT
INI
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Updater_Structure
Update Line Format

FONT

FONT

Copy and install a font in the system.

Windows systems only, not available on Linux

Parameter 1: The name of the font file.

Parameter 2: [FONT]

Parameter 3: The name of the font typeface.

Example:

```
Handg___.ttf,[FONT],HandelGothic
```

Back to the Installation Script Index

See Also

Updater Packaging Introduction
Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
Copy
DELALL
DELFILES
File_Locations_-_Directory_Standards
INI
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Updater_Structure
Update Line Format

Home > Updater > INI

INI

INI

Modify (replace) or add a new key to an INI file.

Parameter 1: The INI file section name to be modified/added to.

Parameter 2: [INI]

Parameter 3: (not used)

Parameter 4: The KEY to add a value to in the INI file.

Parameter 5: The value to add to the key. See the note above about commas.

Parameter 6: (optional): If provided, the name of the INI file in the CONFIG directory to modify. If this parameter is not provided, SETTINGS.INI is assumed.

Examples:

```
hspi_HAI,[INI],xxx,MyText,The only text for this key,MyFile.ini
```

(This would write The only text for this key as the value of the MyText key in the hspi_HAI section of the MyFile.ini file.

Back to the Installation Script Index

See Also

Updater Packaging Introduction
Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
Copy
DELALL
DELFILES
File_Locations_-_Directory_Standards
FONT
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Updater_Structure
Update Line Format

Home > Updater > INIADD

INIADD

INIADD

Add to an existing INI key entry.

Parameter 1: The INI file section name to be modified/added to.

Parameter 2: [INIADD]

Parameter 3: (not used)

Parameter 4: The KEY to add a value to in the INI file.

Parameter 5: The value to add to the key. See the note above about commas.

Parameter 6: (optional): If provided, the name of the INI file in the CONFIG directory to modify. If this parameter is not provided, SETTINGS.INI is assumed.

Examples:

hspi_HAI,[INIADD],xxx,MyText,even more text to add to this entry,MyFile.ini

(This would add even more text to add to this entry to whatever was already set as the value of the MyText key in the hspi_HAI section of the MyFile.ini file.

Back to the Installation Script Index

See Also

Home > Updater > INIADDPARM

INIADDPARM

INIADDPARM

Add a parameter to an existing INI key entry. (Comma separated)

Parameter 1: The INI file section name to be modified/added to.

Parameter 2: [INIADDPARM]

Parameter 3: (not used)

Parameter 4: The KEY to add a value to in the INI file.

Parameter 5: The value to add to the key.

Parameter 6: (optional): If provided, the name of the INI file in the CONFIG directory to modify. If this parameter is not provided, SETTINGS.INI is assumed.

Examples:

```
settings,[INIADDPARM],,io_interfaces,test plugin
```

(This would add test plugin to whatever was already set as the value of the io_interfaces key in the settings section of the settings.ini file. If there is a value already present, then a comma will be added before the new value is appended. If the value is empty or does not exist, then the new value will be set for the key.

Back to the Installation Script Index

See Also

Home > Updater > INSTALLER

INSTALLER

INSTALLER

Run a 3rd party installer program with optional parameters.

Parameter 1: Name of the installer executable.

Parameter 2: [INSTALLER]

Parameter 3: Optional parameters NOT separated by commas.

Examples:

CreateShortcut.exe,[INSTALLER],%HSPATH%\BIN\NEO Editor\Prog.exe

The program will be run with the command line parameters (Parameter 3) appended after it. If you use %HSPATH% in your parameter 3 text, then the updater will dynamically replace that with the root path of HomeSeer (usually C:\Program Files\Homeseer).

Back to the Installation Script Index

See Also

ISINSTALLER

ISINSTALLER

Run a 3rd party InstallShield installer program.

Parameter 1: Name of the installer executable.

Parameter 2: [ISINSTALLER]

Examples:

MySetup.exe,[ISINSTALLER]

The program will be run with the command line parameters /s /v/qn If your installer is not an InstallShield compatible installer, please make sure these parameters will not have any adverse affects on its operation.

Back to the Installation Script Index

See Also

LOCALCOPY

LOCALCOPY

Copy a local file from one place to another under the HomeSeer directory structure. (This is useful for making backup copies of existing files when updated versions are being installed.)

Parameter 1: Source file name

Parameter 2: [LOCALCOPY]

Parameter 3: Destination file name

Parameter 4: (optional): File copy attributes as for normal installer copy operations minus the file delete attribute.

Source and Destination file names are referenced from the HomeSeer root directory WITHOUT the leading .\ as is used in other updater commands. Destination directories that do not exist will be created.

Example:

```
html\TouchPad\Button.gif,[LOCALCOPY],html\TouchPad\Saved\Button.gif,16
```

Back to the Installation Script Index

See Also

LOCALCOPYNONFATAL

LOCALCOPYNONFATAL

Set the option in the updater for whether or not errors with the LOCALCOPY function will terminate the installation script.

Parameter 1: (not used)

Parameter 2: [LOCALCOPYNONFATAL]

Parameter 3: True or False

Example:

```
,[LOCALCOPYNONFATAL],True
```

The above sets the option to True, so copy errors will not abort the installation.

Back to the Installation Script Index

See Also

Testing Your Package

Testing Your Package

After you have created an updater package and you wish to test it before sending it to HomeSeer Technologies, you can follow these instructions.

First, make sure you have read over the Updater Structure section.

Next, take your updater control line for your package and copy it into the file updater_override.txt in the HomeSeer folder directory. You can review other package control lines in the updater.txt file in your HomeSeer root directory and edit one that looks similar to your type of package to create a control line for your plug-in for testing purposes. HST will create the control line in the production updater control file. Contact HST if you need help creating your control line for testing purposes.

The updater_override.txt file needs some information at the top of the file. Here is a sample file, replace the Z-Wave line with your information:

```
'Name, Updater version, File version, Notices version
HomeSeer Updates File, 3.0.0.0, 14, 1
.,anonymous,user@updates,.,,,0.0.0.0
'%<HS TYPE=SUBCAT>Updates
'%<HS TYPE=SUBCAT>Local Updates
Z-Wave, HSPI_ZWave.exe, ,1.0.0.0, 31, 1, local,ZIPS, HSPI_ZWave_1.0.0.0.zip, 4, Free, HomeSeer Tech, , http://homeseer.com/updates3/icons/Z-Wave.bmp, http://homeseer.com/updates3/descriptions/Z-Wave.htm,,
```

Place your package installation ZIP file into the \Updates3\Zips folder. The file must be named the exact same as the name referenced in the control line you inserted into updater_override.txt.

Now go to the menu Plugins->Manage and click on the Refresh button so it finds your updater_override.txt file and it should list your package.

See Also

Home > Updater > UNZIP

UNZIP

UNZIP

Unzip a zip archive to a directory, do not overwrite existing files.

Parameter 1: The zip archive to process.

Parameter 2: [UNZIP]

Parameter 3: The destination directory for the zip archive. Note that if the archive was created with directory information, then it will be unzipped starting at (destination directory) as the root and will be unzipped with the directory structure in tact. Files that already exist will NOT be overwritten.

Examples:

```
Webhelp.zip,[UNZIP],.\html\webhelp
```

Back to the Installation Script Index

See Also

Home > Updater > UNZIPOVER

UNZIPOVER

UNZIPOVER

Unzip a zip archive to a directory, overwriting existing files.

Parameter 1: The zip archive to process.

Parameter 2: [UNZIPOVER]

Parameter 3: The destination directory for the zip archive. Note that if the archive was created with directory information, then it will be unzipped starting at (destination directory) as the root and will be unzipped with the directory structure in tact. Files that already exist will NOT be overwritten.

Examples:

```
Webhelp.zip,[UNZIPOVER],.\html\webhelp
```

Back to the Installation Script Index

See Also

Home > Updater > Updater_Structure

Updater_Structure

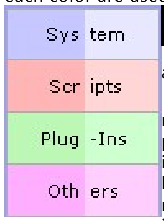
Updater Structure

The HomeSeer updater has the following components and flow:

The updater user interface exists in the HTML\Updater\Updater.aspx ASPX web page. When it is executed, one of the first things it does is to download the latest updater control file, updater.txt, from the HomeSeer server.

After examining the updater.txt control file, the updater determines if there is a newer version of the user notices, as well as the icons and descriptions for packages used by the updater. If either of these are newer on the HomeSeer server, they are downloaded. In the case of a new notices document, a message is displayed the screen immediately after the updater.txt file was downloaded. The user can click on the link provided to view the updated notices. If updated icons and descriptions are downloaded, they are unzipped into their own folder under the HTML\Updates2 folder. (This is also the location of the notices document.) The \Updates2 folder then receives a copy of the update control file named "last_update.txt" and is used for comparison to determine whether notices/icons/descriptions need to be downloaded the next time the updater is run.

After the opportunity to view the notices, the next screen is the list of packages that are presented to the user. The packages are grouped under (currently) three section headings: HomeSeer Technologies Updates, HomeSeer Technologies Plug-Ins and Scripts, and 3rd Party Plug-Ins and Scripts. Under each of these sections are listed the packages, using a color coding scheme to make the package type easier to determine. Two shades of each color are used to offset the rows where two packages are above or below one another. The current color codes are shown here and are:



Blue indicates updates for the HomeSeer system itself, **Red** is used for script packages, **Green** indicates plug-ins, and a pseudo **Light-Violet/Pink** is used to indicate "Other" packages that do not fit into one of the other categories. The updater handles the installation of script and plug-in packages differently since script packages require HomeSeer to be running so that installation scripts can be executed, and plug-ins cannot have HomeSeer fully operational because the plug-in program would not be able to be updated due to its being in use by HomeSeer. Even if a plug-in is not currently enabled, interrogation by HomeSeer at startup can cause plug-in files to be loaded into memory which prevents the file from being able to be replaced. The updater always tries to install the packages that it can install, and so proper coding of the package type is necessary. Packages such as a standalone application may be coded into the "Other" package type so that they are installed immediately instead of when HomeSeer is restarted.

Each package row contains information regarding the package - the information shown is as follows:

- The "Install" column contains a checkbox where the user checks a package they wish to have installed.
- The "Installed" column shows "Yes" when the package is already installed and is the same version as is offered in the updater, "No" if the package is already installed but the updater has a newer version available, "Older" if the version in the updater is older than the one already installed (which happens when users beta test software for the package author) and is blank if the package is not detected as being installed on the HomeSeer computer.

- The icon if shown can help you find/identify the package to the users.
- The package name is self explanatory, and it is hyperlinked to the package description HTM document which is displayed in a new popup window when the link is clicked.
- The type is also pretty self explanatory - it is the type of package being shown.
- The version is the version of the package being offered in the updater.
- The installed version is the version information off of the existing plug-in, or in the case of a script or 'other' package type, it is the version from the package the last time it was installed via the updater.
- The cost column shows the cost or whether the package is Free, but remember that all licensed plug-ins and scripts are allowed a 30 day trial period automatically.
- The provider column lets you know who or which organization is providing the package - this is sometimes a hyperlink to send email or perhaps to a website with more information regarding the provider.

After selecting one or more packages to be installed and clicking the 'Next' button, the choices are confirmed and then if the 'Next' button is pressed again, downloading of the packages begins. The downloading of packages is managed by HomeSeer, not Updater.ASPX. The updater interface copies the control record for the package selected to be downloaded into the DownloadPending.txt file in the \Updates2 directory.

After a package's installation ZIP file has been downloaded, the record is removed from the download pending file and it is placed into the \Updates2\pending.txt file, which is the control file for updates that are to be installed.

When all of the packages are downloaded, the pending.txt file is examined and any packages that are scripts or 'other' type packages are immediately installed. Upon successful installation, the record from the pending.txt file is removed. Plug-in packages remain in the pending.txt file until HomeSeer is restarted.

When HomeSeer starts, the pending.txt file is checked immediately for any records which indicate packages that need to be installed. These are installed and if successful, the record is removed from the pending.txt file. This takes place early in the startup of HomeSeer so that none of the plug-ins are initialized yet, thus facilitating the updater's ability to unregister, replace, and register the plug-in executable software.

See Also

Updater Packaging Introduction
Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
Copy
DELALL
DELFILES
File_Locations_-_Directory_Standards
FONT
INI
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Update Line Format

Update Line Format

An update line is a single line of text that describes your update. It will normally be retrieved from your own web site. The line describes how to get your zip file package, what the version is, etc. Here is a sample of what the line looks like:

```
Sample Plugin, HSPI_SAMPLE.exe, ,1.0.0.0, 63, 1, homeseer.com,/updates3, HSPI_SAMPLE_1_0_0_0.zip, 4, Free,
HomeSeer Tech, Sample Plugin, http://homeseer.com/updates3/icons/Plug-In.gif, sample.htm, 0219,
http://store.homeseer.com/store/X-P219.aspx
```

The above line should be hosted on your own server and special line provider for the HomeSeer updater file. For example, you can use a free drop box account to host your updater line in a file named "acme_product_update.txt". We would then add a line similar to this in the HomeSeer updates.txt file:

@http://dl.dropbox.com/u/52503545/acme_product_update.txt

You can get the URL to your file from within dropbox.

The parts of this line are seperated with a comma and contain the following, in this order:

1. Name of the plugin (a descriptive name that will be displayed in the Updater, keep as short as possible)
2. Name of the file of your plugin. This will be your EXE file. This is case sensitive.
3. The version of your plugin
4. A bit mapped value that describes what products and operating systems your plugin supports, defined as follows:
   bit 1 = supports the standard software
   bit 2 = PRO software
   bit 4 = PRO hardware (PRO-100 unit which runs the PRO software)
   bit 8 = HomeTroller hardware
   bit 16 = Supports Windows OS
   bit 32 = Supports Linx
5. Source of the plugin, 1=HomeSeer developed, 2=3rd Party
6. FTP host where the plugin zip package exists
7. Directory on the server where the zip file exists
8. Filename of zip file package, case sensitive
9. Update type, 4=plugin, 3=script
10. Cost of the package to purchase, if free enter "Free"
11. Provide name of package like company name
12. Registration name of the plugin, this is the name that is returned by the Name property in your plugin
13. URL to your Icon file
14. URL to your information file
15. SKU as given to you by HomeSeer so use in the HomeSeer store
16. URL to more information. This will be a URL to the HomeSeer store product page

See Also

Updater Packaging Introduction
Installation Script Index
ASKY
ASK_or_ASKN
CHECKVERSION
Contents_of_an_Updater_Submission
Copy
DELALL
DELFILES
File_Locations_-_Directory_Standards
FONT
INI
INIADD
INIADDPARM
INSTALLER
ISINSTALLER
LOCALCOPY
LOCALCOPYNONFATAL
Testing Your Package
UNZIP
UNZIPOVER
Updater_Structure

Appendices

Articles in this section

See Also

Document Revisions
Getting Started
Controlling with JSON
Plugin Initialization
Base Plugin API
Devices
Callbacks
Triggers
Actions
Webpages
Speak Proxy
Script ASP
Technology APIs
Updater

Appendix A - Best Practices

Here are a few simple suggestions for making your plug-in work well with HomeSeer, and the user community:

- Do put a logging mechanism in your plug-in. Give the users the ability to turn it on with a simple web interface or script command. Provide an option to write the log to HomeSeer's log or your own file (or both) whichever is more useful to you in diagnosing problems. Plugins that are started in developer mode (check box on interfaces page) will display a console window. You can use Console.WriteLine(info) to display logging information when you need to assist a user in troubleshooting.
- Do not use form windows, MsgBox, or other Windows specific calls.
- Test your plug-in under Linux and ensure it works properly under that OS. By using only .NET functions you can be sure your plugin will support Linux.
- Use thread.sleep(1) in a loop with a timed or conditional exit to release the CPU when you are waiting, this ensures other processes will run and it will not cause the CPU to go to 100%.
- Do not create unnecessary devices without giving the user an option. Many users will choose only a few devices to access the information they want, and will then have to hide unwanted devices. Unwanted devices place an additional burden on your plug-in and HomeSeer to maintain.
- For devices that you do create, use the device values. If you only update the device string value (the display string), the device date/time is not normally updated. If you set a value on the device as well as setting the string value, then the users can have device value triggers AND they can view the last date/time that the device was updated. If you do only update the display string, then use the 'reset' parameter of the SetDeviceString command to update the date/time. For more information on devices within HomeSeer, please see this topic.

See Also

Appendix B - HomeSeer Constants
Appendix C - Useful HomeSeer Settings
Appendix D - Sample Plug-In

Appendix B - HomeSeer Constants

HomeSeer uses constants for many of the device and event properties. For convenience, some of these constants are provided here should you wish to use the same named constants in your plug-in.

User access levels

```
 Public Const USER_GUEST As Integer = 1    ' user can view web pages only, cannot make changes
  Public Const USER_ADMIN As Integer = 2    ' user can make changes
  Public Const USER_LOCAL As Integer = 4    ' this user is used when logging in on a local subnet
  Public Const USER_NORMAL As Integer = 8   ' Not guest, not admin, just NORMAL!
```

Plug-In Capabilities

```
  Public Const CA_IO As Integer = 4        ' supports I/O, must be defined for ALL plugins
  Public Const CA_SEC As Integer = 8       ' security, currently not supporte
  Public Const CA_THERM As Integer = 16    ' Indicates a thermostat plug-in.
  Public CONST CA_MUSIC As Integer = 32    ' Music API, currently not supported
```

Device MISC bit settings

```
 Public Const MISC_NO_LOG As Integer = 8                 ' no logging to event log for this device
  Public Const MISC_STATUS_ONLY As Integer = &H10       ' device cannot be controlled
  Public Const MISC_HIDDEN As Integer = &H20            ' device is hidden from views
  Public Const MISC_INCLUDE_PF As Integer = &H80        ' if set, device's state is restored if power fail
enabled
  Public Const MISC_SHOW_VALUES As Integer = &H100      ' set=display value options in win gui and web status
  Public Const MISC_AUTO_VC As Integer = &H200          ' set=create a voice command for this device
  Public Const MISC_VC_CONFIRM As Integer = &H400       ' set=confirm voice command
  Public Const MISC_SETSTATUS_NOTIFY As Integer = &H4000  ' if set, SetDeviceStatus calls plugin SetIO
  Public Const MISC_SETVALUE_NOTIFY As Integer = &H8000 ' if set, SetDeviceValue calls plugin SetIO
  Public Const MISC_NO_STATUS_TRIG As Integer = &H20000 ' if set, the device will not appear in the device
status
```

HSEvent Callback Types used with RegisterEventCB

```
' For HSEvent callbacks

 Public Const EV_TYPE_X10 As Integer = 1
  Public Const EV_TYPE_LOG As Integer = 2
  Public Const EV_TYPE_STATUS_CHANGE As Integer = 4
  Public Const EV_TYPE_AUDIO As Integer = 8
  Public Const EV_TYPE_X10_TRANSMIT As Integer = &H10S
  Public Const EV_TYPE_CONFIG_CHANGE As Integer = &H20S
  Public Const EV_TYPE_STRING_CHANGE As Integer = &H40S
  Public Const EV_TYPE_SPEAKER_CONNECT As Integer = &H80S
  Public Const EV_TYPE_CALLER_ID As Integer = &H100
  Public Const EV_TYPE_ZWAVE As Integer = &H200
  Public Const EV_TYPE_VALUE_CHANGE As Integer = &H400
  Public Const EV_TYPE_STATUS_ONLY_CHANGE As Integer = &H800
  Public Const EV_TYPE_GENERIC As Integer = &H8000
```

Phone LINEStatus Values

```
 Public Const LINE_IDLE As Integer = 0
  Public Const LINE_OFFERING As Integer = 1
  Public Const LINE_RINGING As Integer = 2
  Public Const LINE_CONNECTED As Integer = 3
  Public Const LINE_INACTIVE As Integer = 4
  Public Const LINE_BUSY As Integer = 5
  Public Const LINE_INUSE As Integer = 6
  Public Const LINE_TIMEOUT As Integer = 7 ' for calling
```

See Also

Appendix A - Best Practices
Appendix C - Useful HomeSeer Settings
Appendix D - Sample Plug-In

Home > Appendices > Appendix C - Useful HomeSeer Settings

Appendix C - Useful HomeSeer Settings

There are several settings in the HomeSeer Settings.INI file that you may wish to use in your plug-in. Some of the more useful ones are listed here. Unless you understand fully the ramifications, it is not recommended that you CHANGE any of the settings through the use of the INI script commands.

Under [Settings]

app_path - The path to where HomeSeer is installed

gLogDir - If True, the HomeSeer log file is kept in the Logs directory instead of the root directory.

logenable - If '1', then logging is enabled.

log_mem - If '1', then logging to the log screen is enabled.

gLogName - Lets you know if the log file is HomeSeer.log or Ah.log

auto_refresh - If greater than 0, then certain web pages are refreshed at this interval (seconds).

gNavigationLoc - top/bottom/both, denotes the location of the navigation bar links.

UseLocation2 - If True, the system uses the Location2 property on devices.

bLocationFirst - If True, then the use of location and location2 is that location comes first in describing devices.

The following are especially useful for launching web pages within the HS system:

svrport - The HomeSeer web server port number.

gServerAddressBind - If it is not 0.0.0.0, then the web server was bound to a specific IP address.

gUseIEBrowser - If False, the user uses a browser registered to handle web pages that may not be IE.

gWebSvrSSLEnabled - If True, the SSL web server is enabled.

gWebSvrSSLPort - The port number the SSL server uses, if enabled.

See Also

Appendix A - Best Practices
Appendix B - HomeSeer Constants
Appendix D - Sample Plug-In

Home > Appendices > Appendix D - Sample Plug-In

Appendix D - Sample Plug-In

The latest sample plugins are posted to our message board at forums.homeseer.com. See the HS3 Plugin Development forum.

See Also

Appendix A - Best Practices
Appendix B - HomeSeer Constants
Appendix C - Useful HomeSeer Settings

Index