

# **Is Speaking Plug-In for HomeSeer**

## **User's Guide**

*Plug-In Version 3.0*

**Thursday, January 18, 2007**

[Click HERE to return to HomeSeer](#)

## Table of Contents

[Introduction](#)

[Speak Proxy Function](#)

[Script Commands](#)

[GetNextQueueItem](#)

[GetNextQueueData](#)

[GetQueueCount](#)

[GetAllQueueData](#)

[RemoveFromQueue](#)

[StopSpeakProxy](#)

[StartSpeakProxy](#)

[Optional INI File Settings](#)

---

Welcome!

This plug-in provides your HomeSeer environment with the ability to know when HomeSeer is speaking something, which is especially useful when you are trying to control some speaker switching events.

This plug-in has the following features:

- Three devices are created if they do not already exist, and they are called “Is Speaking Status”, “Is Speaking Speak-In”, and “Is Speaking Speak-Out” and they are set to the location of “Is Speaking”. Once these devices are created, you may change their location as you see fit. The status device reflects whether HomeSeer is speaking or not. It has a status/value pair that indicates numerically when it is speaking (value = 1) and visually with a status display of “Is Speaking”. You can use this as a trigger in HomeSeer by using the “Device Value Change” trigger. This same device also changes to let you know when HomeSeer is no longer speaking with a value of 0, and a status display of “Not Speaking”.
- A new condition appears in HomeSeer so that you can check to see if HomeSeer is speaking or not. Since HomeSeer’s speaking is usually for a short time, this condition should not be used by itself to trigger an event, but it is very useful for determining if HomeSeer is speaking when used with another kind of trigger. The value of this condition is “Is Speaking” or “Is Not Speaking”.
- A script command exists that will allow you to suspend your scripts or events until HomeSeer is finished speaking. The script command is formatted in Version 2 script command syntax and is called ContinueAfterSpeaking. The format for using this command in a script is as follows:  
`hs.plugin("Is Speaking").ContinueAfterSpeaking`

Because of how HomeSeer processes multiple scripts/events at a time, you may not be able to use “Is Speaking” to switch multiple speakers on and off at the start of HomeSeer speaking. You can, however, use “Is Speaking” to let you know when it is finished speaking so that speakers or other devices can be returned to normal as soon as speaking is complete. The full [speak proxy](#) device handshaking feature will have to be used if you wish to turn equipment on/off before or after speaking.

**Important Notes:** The state of Is Speaking reflects HomeSeer playing audio files as well as speaking, so feel free to use Is Speaking to aid you in your control of WAV files being played too. Please keep in mind also that media files played with the Media Player plug-in are played using Microsoft's Windows Media Player, not HomeSeer, so this plug-in does NOT reflect audio files being played with that plug-in.

**When Is Speaking is enabled for the first time, and if an IsSpeaking.ini file does not exist in the \Config folder of HomeSeer, then a fourth device "Is Speaking Documentation" will be created. The status of the device is a hyperlink that will direct the browser to the documentation you are now reading. Once the device is created, an entry is placed in the IsSpeaking.ini file so that the file exists, which means that removing the documentation device will not cause it to be re-created.**

Here is an example of the ContinueAfterSpeaking script command and how it can be used to synchronize speaker control in a script:

```
Sub Main ()

    hs.run "Speaker_Save_State.txt"    ` Saves the current state of the talkback
speakers
    hs.run "Speakers_Only_Office.txt" ` Turn on only the office speaker
    hs.run "Speak_Appointments.txt"  ` Speak the day's appointments
    Do while hs.IsScriptRunning("Speak_Appointments.txt")
        hs.WaitEvents
    Loop
    hs.plugin("Is Speaking").ContinueAfterSpeaking    ` Wait for speaking to end -
even
                                                    ` if the script is done,
speaking
                                                    ` may not be done.

    hs.run "Speaker_Restore_State.txt"

End Sub
```

[Return to Table of Contents](#)

## ***Using IsSpeaking as a Speak Proxy for HomeSeer***

In HomeSeer version 1.6.169, support was added to HomeSeer for a plug-in to register itself as a speak proxy, which means that requests to speak is handed to a plug-in, and it is then the responsibility of the plug-in to return the text to HomeSeer when it is to be spoken for real.

IsSpeaking 2.x-3.x was written to support this new functionality in addition to it operating as it always has.

The speak proxy support is described in the simplest of terms as: A device is turned ON when HomeSeer tries to speak something, and when you are ready for it to be spoken for real, turn on another device.

Here is the detail:

When HomeSeer or a script in HomeSeer attempts to speak, the plug-in gets this information and stores it in a queue. It immediately turns ON the device “Is Speaking Speak-In”. The value of this device also reflects the current number of items waiting to be spoken in the queue. If the value of this device is 6, then there are 6 calls that were made for HomeSeer to speak that are currently stored in the queue. Use the device status trigger to monitor for this ON status to know when to launch your own custom scripts or events that will do what you want done before you are ready to speak something. You can also set up an event to trigger when the Speak-In device reflects a specific number of messages waiting to be spoken. Using the HomeSeer Device Value Change trigger, you can specify values indicating how many messages are in the queue, and your event will trigger at those value levels. For example, if you set up an event to trigger when the VALUE of Speak-In is Greater Than 4, then the event will trigger when there are 5, 6, 7, 8, etc. speak phrases in the queue.

When you are ready for HomeSeer to speak what is in the queue, turn ON the device “Is Speaking Speak-Out”. HomeSeer will speak everything in the queue, and when it is done, it will turn off both the Speak-In and Speak-Out devices, and the value of the Speak-In device will be set to zero. This means that the HomeSeer Device Status Change trigger can be used with the Speak-Out device to know when the queue is emptied and speaking is done.

There are script commands that you may use to enhance your use of the new features of Is Speaking and they are listed next.

**Note:** Starting with version 2.3.0.0 of IsSpeaking, the plug-in will NOT register itself as a speak proxy automatically when it starts. You must therefore use the StartSpeakProxy script command to enable speak proxying! It is suggested that you add this command to your HomeSeer startup script:

```
hs.PlugIn("Is Speaking").StartSpeakProxy
```

[Return to Table of Contents](#)

## ***New Speak Proxy Script Commands***

New for the 2.x version of Is Speaking is the speak proxy support, and with this comes HomeSeer script commands to access or delete the items to be spoken. They are:

### **Retrieve the oldest (first) text to be spoken in the queue:**

#### **GetNextQueueItem**

Returns: String

Example:

```
hs.writelog "TEST", "Next item is " & hs.PlugIn("Is Speaking").GetNextQueueItem
```

[Return to Table of Contents](#)

### **Retrieve the oldest (first) speak data to be spoken in the queue: (Speak Data includes the device ID and the Wait parameter value in addition to the text to be spoken)**

#### **GetNextQueueData**

Returns: Object Array

Example:

```
sub main()  
    dim sDATA  
    sDATA = hs.PlugIn("Is Speaking").GetNextQueueData  
    hs.writelog "TEST", "Device is " & cstr(sDATA(0))  
    hs.writelog "TEST", "Text is " & sDATA(1)  
    hs.writelog "TEST", "Wait is " & cstr(cbool(sDATA(2)))  
end sub
```

[Return to Table of Contents](#)

### **Retrieve the number of items waiting in the queue:**

#### **GetQueueCount**

Returns: Integer

Example:

```
hs.writelog "TEST", "Queue Count Is:" & cstr(hs.PlugIn("Is Speaking").GetQueueCount)
```

[Return to Table of Contents](#)

### **Retrieve all of the speak data items in the queue:**

## GetAllQueueData

Returns: Two-Dimensional Object Array

Example:

```
sub main()
    dim sDATA
    sDATA = hs.PlugIn("Is Speaking").GetAllQueueData
    for x = 0 to ubound(sDATA)
        hs.writelog "TEST", "Device is " & cstr(sDATA(x,0))
        hs.writelog "TEST", "Text is " & sDATA(x,1)
        hs.writelog "TEST", "Wait is " & cstr(cbool(sDATA(x,2)))
    next
end sub
```

[Return to Table of Contents](#)

## Remove an item from the queue:

### RemoveFromQueue(item as Integer)

Returns: Boolean – True if successful, False if not.

Example:

```
hs.writelog "TEST", "Removing item 1, result is " & cstr(hs.PlugIn("Is Speaking").RemoveFromQueue(1))
```

Note: Text is spoken from the top (position 1) of the queue to the bottom (position n) of the queue. The first (oldest) position in the queue is number 1, not zero. As text is spoken and removed from the queue, the remaining items are reordered and you will have a new number 1 item. Thus, caution should be taken to insure that the queue has not changed before you call RemoveFromQueue.

[Return to Table of Contents](#)

## Turn off the Speak Proxy Feature:

### StopSpeakProxy

Returns: n/a

Example:

```
&hs.PlugIn("Is Speaking").StopSpeakProxy
```

[Return to Table of Contents](#)

## Start or Re-enable the Speak Proxy Feature:

### StartSpeakProxy

Returns: n/a

Example:

```
&hs.PlugIn("Is Speaking").StartSpeakProxy
```

[Return to Table of Contents](#)

## **Special *IsSpeaking.INI* Values to Control *IsSpeaking* Behavior**

There are three keys for *IsSpeaking* that you can set in the *IsSpeaking.INI* file. If this file does not already have a section named [*IsSpeaking*], you may create it for the purposes of using these settings:

### **[*IsSpeaking*]**

*Debug*=True  
*SpeakStartup*=False  
*InitialProxy*=False

#### ***Debug*=True/False**

This determines whether *IsSpeaking* should write debug information to the HomeSeer log. This is very verbose output, and thus should not be used unless you are working with the author to troubleshoot an issue.

#### ***SpeakStartup*=True/False**

When set to TRUE, the *IsSpeaking* plug-in will not proxy speak requests while the HomeSeer startup is active. This means that any speak events during startup, such as “Welcome to HomeSeer” will not be queued and will be spoken immediately. The “Is Speaking Status” device behavior is unchanged and may still be used to monitor this speaking. When HomeSeer’s startup is complete, *IsSpeaking* will begin to queue speak events.

#### ***InitialProxy*=True/False**

Normally *IsSpeaking* starts with the speak proxy disabled, and you need to add *StartSpeakProxy* to your HomeSeer startup script to enable speak proxying. You can set this value to TRUE and the plug-in will start proxying speak phrases immediately.

[Return to Table of Contents](#)