

# Programming the Microsoft Agent Control

---

## ActiveX™ Technology for Interactive Software Agents



Last updated: November 1997  
Microsoft Corporation

**Note:** This document is provided for informational purposes only and Microsoft makes no warranties, either expressed or implied, in this document. The entire risk of the use or the results of this document remains with the user.

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property rights. Microsoft, MS, MS-DOS, Windows, Windows NT, and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the U.S. and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

## Contents

- Introduction
- Document Conventions
- Accessing the Control in Web Pages
- Accessing the Control from Visual Basic and Other Programming Languages
- Accessing Microsoft Agent Services Directly
- The Request Object
- The Agent Control
- The Characters Object
- The AudioOutput Object
- The SpeechInput Object
- The CommandsWindow Object
- The PropertySheet Object

## Introduction

Although applications can write directly to the Microsoft Agent services using its automation server interfaces, Microsoft Agent also includes an ActiveX™ (OLE) control. The control supports easy programming using a scripting language such as Microsoft® Visual Basic® Scripting Edition (VBScript) or other languages that support the ActiveX control interface.

## Document Conventions

This documentation uses the following typographical conventions:

Convention	Description
<b>Sub, Visible, Caption</b>	Words in bold with initial letter capitalized indicate keywords.
<i>agent, String, Now</i>	Italic words indicate placeholders for information you supply.
ENTER, F1	Words in all capital letters indicate filenames, key names, and key sequences.
<code>Agent1.Commands.Enabled = True</code>	Words in this font indicate code samples.
<code>` This is a comment</code>	An apostrophe ( ` ) indicates a code comment.
<code>Agent1.Commands.Add "Test1", _ "Test 1", "test one"</code>	A space and an underscore ( _ ) continues a line of code.
<code>[words or expression]</code>	Items inside square brackets are optional.
<code>This   That</code>	A vertical bar indicates a choice between two or more items.
<i>agent</i>	The word "agent" in italics represents the name of the agent control you use.

The descriptions of programming interfaces in this document follow the conventions for Microsoft VBScript. However, they should be generally applicable to other languages as well.

## Accessing the Control in Web Pages

To access the Microsoft Agent services from a Web page, use the HTML <OBJECT> tag within the <HEAD> or <BODY> element of the page, specifying the Microsoft CLSID (class identifier) for the control. In addition, use a CODEBASE parameter to specify the location of the Microsoft Agent installation file and its version number. The following example illustrates how to use the CODEBASE parameter to autodownload the English language version 1.5 of Microsoft Agent. For information about the current location of the Microsoft Agent installation file for specific language versions and the current release number available, see the Microsoft Agent download page (<http://www.microsoft.com/workshop/prog/agent/agentdl.htm>).

```
<OBJECT  
  classid="clsid:F5BE8BD2-7DE6-11D0-91FE-00C04FD701A5"  
  CODEBASE = "http://activex.microsoft.com/controls/agent/MSagent.exe#VERSION=1,5,0,0"  
  id=Agent  
>  
</OBJECT>
```

Before any script on the page can access its services, Microsoft Agent must be installed on the system loading the page. If Microsoft Agent is not installed on the target system, Microsoft Internet Explorer will automatically download the Microsoft Agent server, data provider, and ActiveX control and ask the user whether to proceed with installation. Once installed, these three items do not have to be reinstalled unless the user uninstalls them. To begin using a character, you must also download its data using the **Load** and **Get** methods. For more information about the syntax for loading a character, see the **Load** method.

Note that you can also use the methods, properties, and events exposed by the browser to program the character; for example, to program its reaction to a button click. Consult the documentation for your browser to determine what features it exposes in its scripting model. For the Microsoft Internet Explorer, see the Scripting Object Model.

Supporting Microsoft Agent from a Web page requires Microsoft Internet Explorer version 3.0 or later. For other browsers, contact the supplier for information regarding their support for ActiveX.

## Using VBScript

To program Microsoft Agent from VBScript, use the HTML <SCRIPT> tags. To access the programming interface, use the name of control you assign in the <OBJECT> tag, followed by the subobject (if any), the name of the method or property, and any parameters or values supported by the method or property:

```
agent[.object].Method parameter, [parameter]
agent[.object].Property = value
```

For events, include the name of the control followed by the name of the event and any parameters:

```
Sub agent_event (ByVal parameter[,ByVal parameter])
statements...
End Sub
```

You can also specify an event handler using the <SCRIPT> tag's **For...Event** syntax:

```
<SCRIPT LANGUAGE=VBScript For=agent Event=event[(parameter[,parameter])]>
statements...
</SCRIPT>
```

Although Microsoft Internet Explorer supports this latter syntax, not all browsers do. For compatibility, use only the former syntax for events.

With VBScript 2.0, you can verify whether Microsoft Agent is installed by trying to create the object and checking to see if it exists. The following sample demonstrates how to check for the Agent control without triggering an auto-download of the control (as would happen if you included an <OBJECT> tag for the control on the page):

```
<!-- WARNING - This code requires VBScript 2.0.
It will always fail to detect the Agent control
in VbScript 1.x, because CreateObject doesn't work.
-->

<SCRIPT LANGUAGE=VBSCRIPT>
If HaveAgent() Then
    'Microsoft Agent control was found.
```

```

document.write "<H2 align=center>Found</H2>"
Else
    'Microsoft Agent control was not found.
document.write "<H2 align=center>Not Found</H2>"
End If

Function HaveAgent()
' This procedure attempts to create an Agent Control object.
' If it succeeds, it returns True.
' This means the control is available on the client.
' If it fails, it returns False.
' This means the control hasn't been installed on the client.

    Dim agent
    HaveAgent = False
    On Error Resume Next
    Set agent = CreateObject("Agent.Control.1")
    HaveAgent = IsObject(agent)

End Function

</SCRIPT>

```

You can download VBScript 2.0 and obtain further information on VBScript at the Microsoft Download site and the Microsoft VBScript site.

## Using JavaScript and JScript

If you use JavaScript™ or Microsoft JScript™ to access Microsoft Agent's programming interface, follow the conventions for this language for specifying methods or properties:

```

agent.object.Method (parameter)
agent.object.Property = value

```

JavaScript does not currently have event syntax for non-HTML objects. However, with Internet Explorer you can use the <SCRIPT> tag's **For...Event** syntax:

```

<SCRIPT LANGUAGE="JScript" FOR="object" EVENT="event()" ">
statements...
</SCRIPT>

```

Because not all browsers currently support this event syntax, you may want to use JavaScript only for pages that support Microsoft Internet Explorer or for code that does not require event handling.

The current release of JScript does not support collections. To access methods and properties of the Character object, use the Character method. Similarly, to access the properties of a Command object, use the Command method.

## Accessing Speech Support for Microsoft Agent

Although Microsoft Agent's services include support for speech recognition, a compatible speech engine must be installed to access Agent's speech recognition support. Your license for Microsoft Agent includes a license for the Microsoft Command and Control speech recognition engine when used as part of a Microsoft Agent client application.

To support automatic downloading and installation of Microsoft Command and Control from an HTML page, include a separate <OBJECT> tag specifying the CLSID of the engine. In addition, include a CODEBASE parameter to specify the location of the installation file as well as the version number, as shown in the following example. For the current location and version number to use for autodownloading the Microsoft Command and Control speech engine, consult the information posted on the Microsoft Agent download site at <http://www.microsoft.com/workshop/prog/agent/agentdl.htm>.

```
<OBJECT
classid="clsid:161FA781-A52C-11d0-8D7C-00A0C9034A7E"
CODEBASE = http://www.research.microsoft.com/research/srg/actcnc.exe#VERSION=3,0,0,1831
>
</OBJECT>
```

The license for Microsoft Command and Control does not permit redistribution of the speech engine independently. For information on licensing the engine separately from Microsoft Agent, contact the Microsoft Speech Research group e-mail alias: [voicebug@microsoft.com](mailto:voicebug@microsoft.com). The Microsoft Command and Control engine is currently available only for English language input; however, other speech recognition vendors supporting the Microsoft Speech API may provide support for other languages. If you use another speech engine, contact its vendor for compatibility, installation, and licensing information.

Similarly, if you want to use Microsoft Agent's synthesized speech services, you must install a compatible text-to-speech (TTS) speech engine for your character's output. Your license for Microsoft Agent includes a license to use a special version of the Lernout & Hauspie® TruVoice engine, but only when used with Microsoft Agent.

To support automatic downloading and installation of this engine from an HTML page, include the engine's CLSID in the <OBJECT> tag. In addition, include a CODEBASE parameter to specify the location of the installation file as well as the version number, such as shown in the following example. For the current location and version number to use for autodownloading the Lernout & Hauspie Text to Speech Engine for Microsoft Agent, consult the information posted on the Microsoft Agent download site at <http://www.microsoft.com/workshop/prog/agent/agentdl.htm>.

```
<OBJECT
classid="clsid:B8F2846E-CE36-11D0-AC83-00C04FD97575"
CODEBASE = "http://activex.microsoft.com/controls/agent/cgram.exe#VERSION=1,5,0,0"
>
</OBJECT>
```

This speech output engine supports only English language output. However, because Microsoft Agent uses the Microsoft Speech API, other languages may be available. If you want to use another compatible speech engine, contact its vendor for further information about their installation and licensing. Note that if you want to download sound (.WAV) files for your character's voice output, you do not have to install a TTS engine.

## Accessing the Control from Visual Basic and Other Programming Languages

You can also use Microsoft Agent's control from Visual Basic® and other programming languages. Make sure that the language fully supports the ActiveX control interface, and follow its conventions for adding and accessing ActiveX controls.

Accessing Microsoft Agent's control from Visual Basic requires that you first create the control. The easiest way to do this is to place an instance of the control on a form. (You may have to add the control to your toolbox before adding it to your form.) Follow Visual Basic syntax for specifying methods, properties, or events. Using Microsoft Agent's control with Visual Basic is very similar to using the control with VBScript, except that events in Visual Basic must include the data type of passed parameters. However, adding the Microsoft Agent control to a form will automatically include Microsoft Agent's events with their appropriate parameters. For more advanced scenarios, such as creating an Agent control at run time, see the **Connected** property.

Most programming languages that support ActiveX controls follow conventions similar to Visual Basic. For programming languages that do not support object collections, you can use the **Character** method and **Command** method to access methods and properties of items in the collection.

## Accessing Microsoft Agent Services Directly

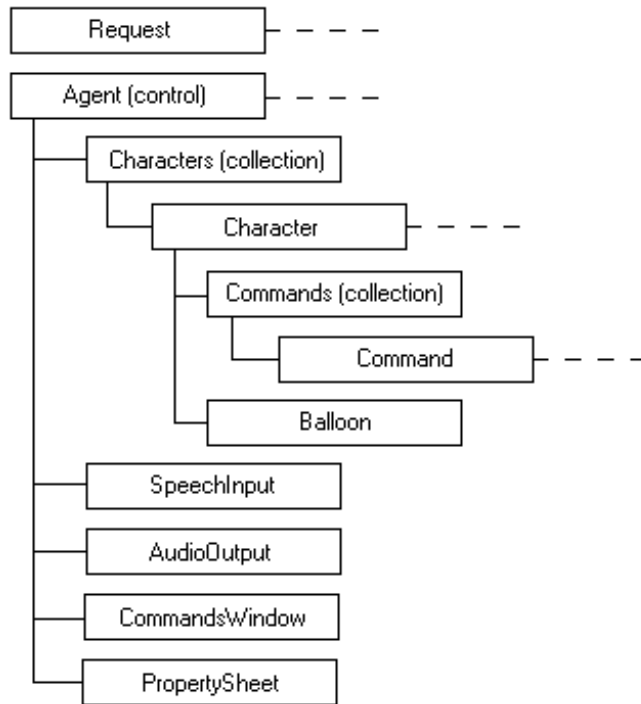
If you are using C, C++, or Java™, you can access the Microsoft Agent server directly using its ActiveX (OLE) interfaces. For more information on these interfaces, see Programming the Microsoft Agent Server Interface.

## The Agent Object Model

The Microsoft Agent Object Model consists of the following objects:

- Request
- Agent (control)
- Characters (collection)
- Character
- Commands (collection)
- Command
- Balloon
- SpeechInput
- AudioOutput
- CommandsWindow
- PropertySheet

These objects are organized in the following hierarchy. (The dotted line following an object indicates that multiple objects can exist.)



**Figure 3. The Agent Object**

## The Request Object

The server processes some methods, such as **Load**, **Get**, **Play**, and **Speak**, asynchronously. This enables your application code to continue while the method is completing. When a client application calls one of these methods, the control creates and returns a **Request** object for the request. You can use the **Request** object to track the status of the method by assigning an object variable to the method. In VBScript and Visual Basic, first declare an object variable:

```
Dim MyRequest as Object
```

In VBScript, you don't include the variable type in your declaration:

```
Dim MyRequest
```

And use Visual Basic's Set statement to assign the variable to the method call:

```
Set Request = agent.Characters("CharacterID").method (parameter[s])
```

This adds a reference to the **Request** object. The **Request** object will be destroyed when there are no more references to it. Where you declare the **Request** object and how you use it determines its lifetime. If the object is declared local to a subroutine or function, it will be destroyed when it goes out of scope; that is, when the subroutine or function is complete. If the object is declared globally, it will not be destroyed until either the program terminates or a new value (or a value set to "empty") is assigned to the object.

The **Request** object provides several properties you can query. For example, the **Status** property returns the current status of the request. You can use this property to check the status of your request:

```

Dim MyRequest

Set MyRequest = Agent1.Characters.Load ("Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf")

If (MyRequest.Status = Pending) then
    'do something

Else If (MyRequest.Status = Complete) then
    'do something right away

End If

```

The **Status** property returns the status of a **Request** object as a Long integer value.

Status	Definition
0	Request successfully completed.
1	Request failed.
2	Request pending (in the queue, but not complete).
3	Request interrupted.
4	Request in progress.

The **Request** object also includes a Long integer value in the **Number** property that returns the error or cause of the **Status** code. If none, this value is zero (0). The **Description** property contains a string value that corresponds to the error number. If the string doesn't exist, **Description** contains "Application-defined or object-defined error".

For the values and meaning returned by the **Number** property, see **Error Codes**.

The server places animation requests in the specified character's queue. This enables the server to play the animation on a separate thread, and your application's code can continue while animations play. If you create a **Request** object reference, the server automatically notifies you when an animation request has started or completed through the **RequestStart** and **RequestComplete** events. Because methods that return **Request** objects are asynchronous and may not complete during the scope of the calling function, declare your reference to the **Request** object globally.

The following methods can be used to return a **Request** object: **GestureAt**, **Get**, **Hide**, **Interrupt**, **Load**, **MoveTo**, **Play**, **Show**, **Speak**, **Wait**.

-----

## The Agent Control

Referencing the Agent control provides access to events and most other objects supported by Microsoft Agent. The Agent control also directly exposes its own set of properties.

### Agent Control Properties



The following properties are directly accessed from the Agent control:

### **Connected, Name, Suspended**

In addition, some programming environments may assign additional design-time or run-time properties. For example, Visual Basic adds **Left**, **Index**, **Tag**, and **Top** properties that define the location of the control on a form even though the control does not appear on the form's page at run time.

### **Connected Property**

#### **Description**

Returns or sets whether the current control is connected to the Microsoft Agent server.

#### **Syntax**

*agent*.**Connected** [= *boolean*]

<b>Part</b>	<b>Description</b>
<i>boolean</i>	A Boolean expression specifying whether the control is connected. <b>True</b> The control is connected. <b>False</b> The control is not connected.

#### **Remarks**

In many situations, specifying the control automatically creates a connection with the Microsoft Agent server. For example, specifying the Microsoft Agent control's CLSID in the <OBJECT> tag in a Web page automatically opens a server connection and exiting the page closes the connection. Similarly, for Visual Basic or other languages that enable you to drop a control on a form, running the program automatically opens a connection and exiting the program closes the connection. If the server isn't currently running, it automatically starts.

However, if you want to create an Agent control at run time, you may also need to explicitly open a new connection to the server using the **Connected** property. For example, in Visual Basic you can create an ActiveX object at run time using the Set statement with the **New** keyword (or **CreateObject** function). While this creates the object, it will not create the connection to the server, so you must use the **Connected** property before any code that calls into Microsoft Agent's programming interface, as shown in the following example:

```
` Declare a global variable for the control
Dim MyAgent as Agent

` Create an instance of the control using New
Set MyAgent = New Agent

` Open a connection to the server
MyAgent.Connected = True

` Load a character
```

```
MyAgent.Characters.Load "Genie", "C:\Some Directory\Genie.acs"

` Display the character
MyAgent.Characters("Genie").Show
```

Note that creating a control using this technique does not expose the Agent control's events. In Visual Basic 5.0, you can access the control's events by including the control in your project's references, and use the **WithEvents** keyword in your variable declaration:

```
Dim WithEvents agent as Agent

` Create an instance of the control using New
Set MyAgent = New Agent
```

Using **WithEvents** to create an instance of the Agent control at run time automatically opens the connection with the Microsoft Agent server. Therefore, you don't need to include a **Connected** statement.

You can close your control's connection to the server at run time by setting the **Connected** property to **False**. However, you must first release all references you defined to objects created by the server. In particular, you must release any references you created to character and command objects. In Visual Basic, you can disassociate a reference to an object by setting the reference to **Nothing**:

```
Dim Genie as IAgentCtlCharacter

Sub LoadCharacter

` Load the character into the Characters collection
Agent1.Characters.Load "Genie", _
"C:\Program Files\Microsoft Agent\Characters\Genie.acs"

` Create a reference to the character
Set Genie = Agent1.Characters("Genie")

End Sub

Sub CloseConnection

` Release the reference to the character object
Set Genie = Nothing

` Close the connection with the server
Agent1.Connected = False

End Sub
```

Although you can reopen your connection by resetting the **Connected** property to **True**, not all information established with the server in the original connection will be preserved. For example, if you loaded a character, you will have to reload it again before you can play any of its animations.

Setting the **Connected** property to **False** does not destroy your instance of the control. You must use the syntax supported by your programming language for releasing the object. For example, in Visual Basic, you set the control to **Nothing**:

```
Set Agent1 = Nothing
```

Attempting to query or set the **Connected** property before creating the control will raise an error.

---

## Name Property

### Description

Returns the name used in code to identify the control. This property is read-only at run time.

### Syntax

*agent*.**Name**

### Remarks

In some programming environments such as Visual Basic, adding the control automatically generates a default name for the control that can be changed at design time. For HTML scripts, you can define the name in the <OBJECT> tag. If you define the name, follow the conventions of the programming language for defining object names.

---

## Suspended Property

### Description

Returns a Boolean indicating the Microsoft Agent server operational state.

### Syntax

*agent*.**Suspended**

### Remarks

The **Suspended** property returns **False** when the server is in its normal running state. When the property returns **True**, the server is in its "suspended" state, which indicates that the user shut down the server and implies that no character interaction is desired. Client applications can only read this property, but you can display your own message box suggesting how to restart the server.

### See Also

**Restart** event, **Shutdown** event

---

## Agent Control Events

The Microsoft Agent control provides several events that enable your client application to track the state of the server:

**ActivateInput, BalloonHide, BalloonShow, Bookmark, Click, Command, DbClick, DeactivateInput, DragComplete, DragStart, Hide, IdleComplete, IdleStart, Move, RequestComplete, RequestStart, Restart, Show, Shutdown, Size**

## ActivateInput Event

### Description

Occurs when a client becomes input-active.

### Syntax

Sub *agent*\_**ActivateInput** (**ByVal** *CharacterID*)

Value	Description
<i>CharacterID</i>	Returns the ID of the character through which the client becomes input-active.

### Remarks

The input-active client receives mouse and speech input events supplied by the server. The server sends this event only to the client that becomes input-active.

This event can occur when the user switches to your **Commands** object, for example, by choosing your **Commands** object entry in the Commands Window or in the pop-up menu for a character. It can also occur when the user selects a character (by clicking or speaking its name), when a character becomes visible, and when the character of another client application becomes hidden. You can also call the **Activate** method (with **State** set to 2) to explicitly make the character topmost, which results in your client application becoming input-active and triggers this event. However, this event does not occur if you use the **Activate** method only to specify whether your client is the active client of the character.

### See Also

**DeactivateInput** event, **Activate** method

-----

## BalloonHide Event

### Description

Occurs when a character's word balloon is hidden.

### Syntax

Sub *agent*\_**BalloonHide** (**ByVal** *CharacterID*)

Value	Description
-------	-------------

<i>CharacterID</i>	Returns the ID of the character associated with the word balloon.
--------------------	-------------------------------------------------------------------

#### Remarks

The server sends this event only to the clients of the character (applications that have loaded the character) that uses the word balloon.

#### See Also

**BalloonShow** event

-----

#### BalloonShow Event

#### Description

Occurs when a character's word balloon is shown.

#### Syntax

Sub *agent*\_**BalloonShow** (**ByVal** *CharacterID*)

<b>Value</b>	<b>Description</b>
<i>CharacterID</i>	Returns the ID of the character associated with the word balloon.

#### Remarks

The server sends this event only to the clients of the character (applications that have loaded the character) that uses the word balloon.

#### See Also

**BalloonHide** event

-----

#### Bookmark Event

#### Description

Occurs when a bookmark in a speech text string that your application defined is activated.

#### Syntax

Sub *agent*\_**Bookmark**(**ByVal** *BookmarkID*)

Value	Description
<i>BookmarkID</i>	A Long integer identifying the bookmark number.

#### Remarks

To specify a bookmark event, use the **Speak** method with a **Mrk** tag in your supplied text. For more information about tags, see Speech Output Tags.

-----

#### Click Event

##### Description

Occurs when the user clicks a character.

##### Syntax

**Sub** *agent\_Click* (**ByVal** *CharacterID*, **ByVal** *Button*, **ByVal** *Shift*, **ByVal** *X*, **ByVal** *Y*)

Value	Description
<i>CharacterID</i>	Returns the ID of the clicked character as a string.
<i>Button</i>	Returns an integer that identifies the button that was pressed and released to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
<i>Shift</i>	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
<i>X,Y</i>	Returns an integer that specifies the current location of the mouse pointer. The X and Y values are always expressed in pixels, relative to the upper left corner of the screen.

#### Remarks

This event is sent only to the input-active client of a character. When the user clicks a character with no input-active client, the server sets its last input-active client as the current input-active client, sending the **ActivateInput** event to that client, and then sending the **Click** event.

**Note** Clicking a character does not disable all other character output (all characters). However, pressing the listening hot key does flush the input-active character's output

and triggers the **RequestComplete** event, passing a **Request.Status** that indicates that the client's queue was interrupted.

-----

## Command Event

### Description

Occurs when a (client's) command is chosen by the user.

### Syntax

**Sub** *agent\_Command*(**ByVal** *UserInput*)

Value	Description																						
UserInput	<p>Identifies the <b>Command</b> object returned by the server.</p> <p>The following properties can be accessed from the <b>Command</b> object.</p> <table><tr><td><b>CharacterID</b></td><td>A string value identifying the name (ID) of the character that received the command.</td></tr><tr><td><b>Name</b></td><td>A string value identifying the name (ID) of the command.</td></tr><tr><td><b>Confidence</b></td><td>A Long integer value indicating the confidence scoring for the command.</td></tr><tr><td><b>Voice</b></td><td>A string value identifying the voice text for the command.</td></tr><tr><td><b>Alt1Name</b></td><td>A string value identifying the name of the next (second) best command.</td></tr><tr><td><b>Alt1Confidence</b></td><td>A Long integer value indicating the confidence scoring for the next (second) best command.</td></tr><tr><td><b>Alt1Voice</b></td><td>A string value identifying the voice text for the next best alternative command match.</td></tr><tr><td><b>Alt2Name</b></td><td>A string value identifying the name of third best command match.</td></tr><tr><td><b>Alt2Confidence</b></td><td>A Long integer identifying the confidence scoring for the third best command match.</td></tr><tr><td><b>Alt2Voice</b></td><td>A string value identifying the voice text for the third best command match.</td></tr><tr><td><b>Count</b></td><td>Long integer value indicating the number of alternatives returned.</td></tr></table>	<b>CharacterID</b>	A string value identifying the name (ID) of the character that received the command.	<b>Name</b>	A string value identifying the name (ID) of the command.	<b>Confidence</b>	A Long integer value indicating the confidence scoring for the command.	<b>Voice</b>	A string value identifying the voice text for the command.	<b>Alt1Name</b>	A string value identifying the name of the next (second) best command.	<b>Alt1Confidence</b>	A Long integer value indicating the confidence scoring for the next (second) best command.	<b>Alt1Voice</b>	A string value identifying the voice text for the next best alternative command match.	<b>Alt2Name</b>	A string value identifying the name of third best command match.	<b>Alt2Confidence</b>	A Long integer identifying the confidence scoring for the third best command match.	<b>Alt2Voice</b>	A string value identifying the voice text for the third best command match.	<b>Count</b>	Long integer value indicating the number of alternatives returned.
<b>CharacterID</b>	A string value identifying the name (ID) of the character that received the command.																						
<b>Name</b>	A string value identifying the name (ID) of the command.																						
<b>Confidence</b>	A Long integer value indicating the confidence scoring for the command.																						
<b>Voice</b>	A string value identifying the voice text for the command.																						
<b>Alt1Name</b>	A string value identifying the name of the next (second) best command.																						
<b>Alt1Confidence</b>	A Long integer value indicating the confidence scoring for the next (second) best command.																						
<b>Alt1Voice</b>	A string value identifying the voice text for the next best alternative command match.																						
<b>Alt2Name</b>	A string value identifying the name of third best command match.																						
<b>Alt2Confidence</b>	A Long integer identifying the confidence scoring for the third best command match.																						
<b>Alt2Voice</b>	A string value identifying the voice text for the third best command match.																						
<b>Count</b>	Long integer value indicating the number of alternatives returned.																						

### Remarks

The server notifies you with this event when your application is input-active and the user chooses a command you defined to appear in the Commands Window or character's pop-up menu. The event passes back the number of possible matching commands in **Count** as well as the name, confidence scoring, and voice text for those matches.

If voice input triggers this event, the server returns a string that identifies the best match in the **Name** parameter, and the second- and third-best match in **Alt1Name** and **Alt2Name**. An empty string indicates that the input did not match any command your application defined; for example, it could be one of the server's defined commands. It is also possible that you may get the same command name returned in more than one

entry. **Confidence**, **Alt1Confidence**, and **Alt2Confidence** parameters return the relative scores, in the range of -100 to 100, that are returned by the speech recognition engine for each respective match. **Voice**, **Alt1Voice**, and **Alt2Voice** parameters return the voice text that the speech recognition engine matched for each alternative. If **Count** returns zero (0), the server detected spoken input, but determined that there was no matching command.

If voice input was not the source for the command, for example, if the user selected the command from the character's pop-up menu, the server returns the name (ID) of the command selected in the **Name** property. It also returns the value of the **Confidence** parameter as 100, and the value of the **Voice** parameters as the empty string ("").

**Alt1Name** and **Alt2Name** also return empty strings. **Alt1Confidence** and **Alt2Confidence** return zero (0), and **Alt1Voice** and **Alt2Voice** return empty strings. **Count** returns 1.

**Note** The Microsoft Command and Control speech recognition engine supports returning values in the parameters of this event. If you use Microsoft Agent with another speech engine, check with the supplier to determine whether their engine supports the Microsoft Speech API interface for returning alternatives and confidence scores.

---

## DbClick Event

### Description

Occurs when the user double-clicks a character.

### Syntax

**Sub agent\_DbClick (ByVal CharacterID, ByVal Button, ByVal Shift, ByVal X, ByVal Y)**

Value	Description
<i>CharacterID</i>	Returns the ID of the double-clicked character as a string.
<i>Button</i>	Returns an integer that identifies the button that was pressed and released to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
<i>Shift</i>	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
<i>X,Y</i>	Returns an integer that specifies the current location of the mouse pointer. The X and Y values are always expressed in pixels, relative to the upper left corner of the screen.



### Remarks

This event is sent only to the input-active client of a character. When the user double-clicks a character with no input-active client, the server sets its last input-active client as the current input-active client, sending the **ActivateInput** event to that client, and then sending the **DbClick** event.

-----

## DeactivateInput Event

### Description

Occurs when a client becomes non-input-active.

### Syntax

**Sub** *agent\_DeactivateInput* (**ByVal** *CharacterID*)

Value	Description
<i>CharacterID</i>	Returns the ID of the character that makes the client become non-input-active.

### Remarks

A non-input-active client no longer receives mouse or speech events from the server (unless it becomes input-active again). The server sends this event only to the client that becomes non-input-active. It does not occur when you use the **Activate** method and set the **State** parameter to 0.

This event occurs when your client application is input-active and the user chooses the caption of another client in a character's pop-up menu or the Commands Window. It may also occur when the user selects the name of another character by clicking or speaking. You also get this event when your character is hidden or another character becomes visible.

### See Also

**ActivateInput** event

-----

## DragComplete Event

### Description

Occurs when the user completes dragging a character.

### Syntax

**Sub** *agent\_DragComplete* (**ByVal** *CharacterID*, **ByVal** *Button*, **ByVal** *Shift*, **ByVal** *X*, **ByVal** *Y*)

Value	Description
-------	-------------

<i>CharacterID</i>	Returns the ID of the dragged character as a string.
<i>Button</i>	Returns an integer that identifies the button that was pressed and released to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
<i>Shift</i>	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
<i>X,Y</i>	Returns an integer that specifies the current location of the mouse pointer. The X and Y values are always expressed in pixels, relative to the upper left corner of the screen.

#### Remarks

This event is sent only to the input-active client of a character. When the user drags a character with no input-active client, the server sets its last input-active client as the current input-active client, sending the **ActivateInput** event to that client, and then sending the **DragStart** and **DragComplete** events.

#### See Also

**DragStart** event

-----

### DragStart Event

#### Description

Occurs when the user begins dragging a character.

#### Syntax

**Sub agent\_DragStart (ByVal CharacterID, ByVal Button, ByVal Shift, ByVal X, ByVal Y)**

<b>Value</b>	<b>Description</b>
<i>CharacterID</i>	Returns the ID of the clicked character as a string.
<i>Button</i>	Returns an integer that identifies the button that was pressed and released to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
<i>Shift</i>	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2).

	These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
X,Y	Returns an integer that specifies the current location of the mouse pointer. The X and Y values are always expressed in pixels, relative to the upper left corner of the screen.

#### Remarks

This event is sent only to the input-active client of a character. When the user drags a character with no input-active client, the server sets its last input-active client as the current input-active client, sending the **ActivateInput** event to that client, and then sending the **DragStart** event.

#### See Also

**DragComplete** event

-----

### Hide Event

#### Description

Occurs when a character is hidden.

#### Syntax

**Sub** *agent\_Hide* (**ByVal** *CharacterID*, **ByVal** *Cause*)

Value	Description
<i>CharacterID</i>	Returns the ID of the hidden character as a string.
<i>Cause</i>	Returns a value that indicates what caused the character to hide. <ul style="list-style-type: none"> <li>1 The user hid the character (using the menu or voice command).</li> <li>3 Your client application hid the character.</li> <li>5 Another client application hid the character.</li> </ul>

#### Remarks

The server sends this event to all clients of the character. To query the current state of the character, use the **Visible** property.

#### See Also

**Show** event, **VisibilityCause** property

-----

### IdleComplete Event

**Description**

Occurs when the server ends the **Idling** state of a character.

**Syntax**

**Sub** *agent\_IdleComplete* (**ByVal** *CharacterID*)

Value	Description
<i>CharacterID</i>	Returns the ID of the idling character as a string.

**Remarks**

The server sends this event to all clients of the character.

**See Also**

**IdleStart** event

-----

**IdleStart Event****Description**

Occurs when the server sets a character to the **Idling** state.

**Syntax**

**Sub** *agent\_IdleStart* (**ByVal** *CharacterID*)

Value	Description
<i>CharacterID</i>	Returns the ID of the idling character as a string.

**Remarks**

The server sends this event to all clients of the character.

**See Also**

**IdleComplete** event

-----

**Move Event****Description**

Occurs when a character is moved.

**Syntax**

**Sub** *agent\_Move* (**ByVal** *CharacterID*, **ByVal** *X*, **ByVal** *Y*, **ByVal** *Cause*)

Value	Description
<i>CharacterID</i>	Returns the ID of the character that moved.
<i>X</i>	Returns the x-coordinate (in pixels) of the top edge of character frame's new location as an integer.
<i>Y</i>	Returns the y-coordinate (in pixels) of the left edge of character frame's new location as an integer.
<i>Cause</i>	Returns a value that indicates what caused the character to move. <ul style="list-style-type: none"> <li>1      The user dragged the character.</li> <li>2      Your client application moved the character.</li> <li>3      Another client application moved the character.</li> </ul>

#### Remarks

This event occurs when the user or an application changes the character's position. Coordinates are relevant to the upper left corner of the screen. This event is sent only to the clients of the character (applications that have loaded the character).

#### See Also

**MoveCause** property, **Size** event

-----

### RequestComplete Event

#### Description

Occurs when the server completes a queued request.

#### Syntax

**Sub** *agent\_RequestComplete* (ByVal *Request*)

Value	Description
<i>Request</i>	Returns the <b>Request</b> object.

#### Remarks

This event returns a **Request** object. Because requests are processed asynchronously, you can use this event to determine when the server completes processing a request (such as a **Get**, **Play**, or **Speak** method) to synchronize this event with other actions generated by your application. The server sends the event only to the client that created the reference to the **Request** object and only if you defined a global variable for the request reference:

```

Dim MyRequest
Dim Genie

Sub window_Onload

Agent1.Characters.Load "Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf"

Set Genie = Agent.Characters("Genie")

` This syntax will generate RequestStart and RequestComplete events.
Set MyRequest = Genie.Get("state", "Showing")
` This syntax will not generate RequestStart and RequestComplete events.
Genie.Get "state", "Hiding"

End Sub

Sub Agent1_RequestComplete(ByVal Request)

If Request = MyRequest Then
    Status = "Showing animation is now loaded"

End Sub

```

**Note** In VBScript 1.0, this event fires even if you don't define references to a **Request** object. This has been fixed in VBScript 2.0, which can be downloaded from <http://microsoft.com/msdownload/scripting.htm>.

## See Also

**RequestStart** event

## RequestStart Event

### Description

Occurs when the server begins a queued request.

### Syntax

**Sub** *agent\_RequestStart* (**ByVal** *Request*)

Value	Description
<i>Request</i>	Returns the <b>Request</b> object.

### Remarks

The event returns a **Request** object. Because requests are processed asynchronously, you can use this event to determine when the server begins processing a request (such as a **Get**, **Play**, or **Speak** method) and thereby synchronize this with other actions generated by your application. The event is sent only to the client that created the

reference to the **Request** object and only if you defined a global variable for the request reference:

```
Dim MyRequest
Dim Genie

Sub window_Onload

Agent1.Characters.Load "Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf"

Set Genie = Agent1.Characters("Genie")

` This syntax will generate RequestStart and RequestComplete events.
Set MyRequest = Genie.Get("state", "Showing")

` This syntax will not generate RequestStart and RequestComplete events.
Genie.Get ("state", "Hiding")

End Sub

Sub Agent1_RequestStart(ByVal Request)

If Request = MyRequest Then
    Status = "Loading the Showing animation"

End Sub
```

The **Status** returns 4 (request in progress) for the **Request** object returned.

**Note** In VBScript 1.0, this event fires even if you don't define references to a **Request** object. This has been fixed in VBScript 2.0, which can be downloaded from <http://microsoft.com/msdownload/scripting.htm>.

#### See Also

**RequestComplete** event

---

## Restart Event

### Description

Occurs when the server restarts from its suspended state.

### Syntax

**Sub** *agent\_Restart* ()

### Remarks

The server sends this event to all client applications when the user chooses to restart the server from its suspended state. However, you will not get this event if you close your connection to the server when the server shuts down.

---

## Show Event

**Description**

Occurs when a character is displayed.

**Syntax**

**Sub** *agent\_Show* (**ByVal** *CharacterID*, **ByVal** *Cause*)

Value	Description
<i>CharacterID</i>	Returns the ID of the character shown as a string.
<i>Cause</i>	Returns a value that indicates what caused the character to display.  2      The user showed the character (using the menu or voice command). 4      Your client application showed the character. 6      Another client application showed the character.

**Remarks**

The server sends this event to all clients of the character. To query the current state of the character, use the **Visible** property.

**See Also**

**Hide** event, **VisibilityCause** property

-----

**Shutdown Event****Description**

Occurs when the user explicitly shuts down (exits) Microsoft Agent.

**Syntax**

**Sub** *agent\_Shutdown*()

**Remarks**

When the user explicitly chooses Exit on the pop-up menu on the Microsoft Agent taskbar icon and confirms the choice in the warning message box, the server sends this event to all connected clients and then exits. The server also sets the control's **Connected** property to **False**. Any subsequent calls you make to the server will fail. You may want to handle this error condition.

-----

**Size Event****Description**

Occurs when a character's size changes.

**Syntax**

**Sub** *agent\_Size* (**ByVal** *CharacterID*, **ByVal** *Width*, **ByVal** *Height*)



Value	Description
<i>CharacterID</i>	Returns the ID of the character that moved.
<i>Width</i>	Returns the character frame's new width (in pixels) as an integer.
<i>Height</i>	Returns the character frame's new height (in pixels) as an integer.

#### Remarks

This event occurs when an application changes the size of a character. This event is sent only to the clients of the character (applications that have loaded the character).

#### See Also

**Move** event

### The Characters Object

Your client application can support one or more characters. In addition, you can share a character among several applications. Microsoft Agent defines the **Characters** object as a collection of characters. To access a character, load the character's data into the **Characters** collection and specify that that item in the collection uses the methods and properties supported for that character.

#### Characters Object Methods

The **Characters** object supports methods for accessing, loading, and unloading characters into its collection:

**Character, Load, Unload**

#### Character Method

##### Description

Returns a **Character** object in a **Characters** collection.

##### Syntax

*agent*.**Characters.Character** "*CharacterID*"

##### Remarks

You can use this method to access a **Character** object's methods and properties.

**Note** This method may be required for some programming languages that do not support collections. It is not required for VBScript or Visual Basic. For further information on specifying **Character** methods, see Character Object Methods.

-----

## Load Method

### Description

Loads a character into the **Characters** collection.

### Syntax

*agent*.**Characters.Load** "*CharacterID*", *Provider*

Part	Description
<i>CharacterID</i>	Required. A string value that you will use to refer to the character data to be loaded.
<i>Provider</i>	Required. A variant data type that must be one of the following: <i>Filespec</i> The local file location of the specified character's definition file. <i>URL</i> The HTTP address for the character's definition file. <i>provider</i> An alternate character definition provider (object).

### Remarks

The Microsoft Agent Data Provider supports loading character data stored either as a single structured file (.ACS) with character data and animation data together or as separate character data (.ACF) and animation (.AAF) files. Use the single structured .ACS file to load a character that is stored on a local disk or network and accessed using a conventional file protocol (such as UNC pathnames). Use the separate .ACF and .AAF files when you want to load the animation files individually from a remote site where they are accessed using the HTTP protocol.

For .ACS files, using the **Load** method provides access a character's animations. For .ACF files, you also use the **Get** method to load animation data. The **Load** method does not support downloading .ACS files from an HTTP site.

Loading a character does not automatically display the character. Use the **Show** method first to make the character visible.

If you create an object reference and assign it to this method, it returns a **Request** object. If you use HTTP protocol, assigning a **Request** object to the **Load** method and checking its status in the **RequestComplete** event enables you to prevent your code from failing when the character data fails to load.

The *Provider* parameter also enables you to specify your own data provider (that would be loaded using a separate control) that can have its own methods for loading animation data. You only need to create a data provider object if you supply character data in special formats.

To load a character from the Microsoft Agent site, consult the character data page at <http://www.microsoft.com/workshop/prog/agent/characterdata.htm> for the latest information on the location of the character files.

---

## Unload Method

### Description

Unloads the character data for the specified character.

### Syntax

*agent*.**Characters.Unload** "*CharacterID*"

### Remarks

Use this method when you no longer need a character, to free up memory used to store information about the character. If you access the character again, use the **Load** method.

This method does not return a **Request** object.

---

## Character Object Methods

The server also exposes methods for each character in a **Characters** collection. The following methods are supported:

**Activate, GestureAt, Get, Hide, Interrupt, MoveTo, Play, Show, Speak, Stop, StopAll, Wait**

To use a method, reference the character in the collection. In VBScript and Visual Basic, you do this by specifying the ID for a character:

```
Sub FormLoad

    'Load the genie character into the Characters collection
    Agent1.Characters.Load "Genie", _
        "C:\Program Files\Microsoft Agent\Characters\Genie.acs"

    'Display the character
    Agent1.Characters("Genie").Show
    Agent1.Characters("Genie").Play "Greet"
    Agent1.Characters("Genie").Speak "Hello. "

End Sub
```

To simplify the syntax of your code, you can define an object variable and set it to reference a character object in the **Characters** collection; then you can use your variable to reference methods or properties of the character. The following example demonstrates how you can do this using the Visual Basic Set statement:

```
'Define a global object variable
Dim Genie as Object

Sub FormLoad

    'Load the genie character into the Characters collection
    Agent1.Characters.Load "Genie", _
        "C:\Program Files\Microsoft Agent\Characters\Genie.acs"

    'Create a reference to the character
    Set Genie = Agent1.Characters("Genie")
```

```

'Display the character
Genie.Show

'Get the Restpose animation
Genie.Get "animation", "RestPose"

'Make the character say Hello
Genie.Speak "Hello."

End Sub

```

In Visual Basic 5.0, you can also create your reference by declaring your variable as a **Character** object:

```

Dim Genie as IAgentCtlCharacter

Sub FormLoad

'Load the genie character into the Characters collection
Agent1.Characters.Load "Genie", _
    "C:\Program Files\Microsoft Agent\Characters\Genie.acs"

'Create a reference to the character
Set Genie = Agent1.Characters("Genie")

'Display the character
Genie.Show

End Sub

```

Declaring your object of type **IAgentCtlCharacter** enables early binding on the object, which results in better performance.

In VBScript, you cannot declare a reference as a particular type. However, you can simply declare the variable reference:

```

<!--

Dim Genie

SUB window_OnLoad

'Load the character
AgentCtl.Characters.Load "Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf"

'Create an object reference to the character in the collection
set Genie= AgentCtl.Characters ("Genie")

'Get the Showing state animation
Genie.Get "state", "Showing"

'Display the character
Genie.Show

End Sub

-->

```

```
</SCRIPT>
```

Some programming languages do not support collections. However, you can access a **Character** object's methods with the **Character** method:

```
agent.Characters.Character("CharacterID").method
```

In addition, you can also create a reference to the **Character** object to make your script code easier to follow:

```
<SCRIPT LANGUAGE="JScript" FOR="window" EVENT="onLoad()">
<!--

//Load the character's data
AgentCtl.Characters.Load ("Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf");

//Create a reference to this object
Genie = AgentCtl.Characters.Character("Genie");

//Get the Showing state animation
Genie.Get("state", "Showing");

//Display the character
Genie.Show();

-->
</SCRIPT>
```

-----

## Activate Method

### Description

Sets the active client or character.

### Syntax

**agent.Characters ("CharacterID").Activate [State]**

Part	Description
<i>State</i>	Optional. You can specify the following values for this parameter:  0      Not the active client.  1      The active client.  2      (Default) The topmost character.

### Remarks

When multiple characters are visible, only one of the characters receives speech input at a time. Similarly, when multiple client applications share the same character, only one of the clients receives mouse input (for example, Microsoft Agent control click or drag events). The character set to receive mouse and speech input is the topmost character

and the client that receives the input is the active client of that character. (The topmost character's window also appears at the top of the character windows z-order.) Typically, the user determines the topmost character by explicitly selecting the character. However, topmost activation also changes when a character is shown or hidden (the character becomes or is no longer topmost, respectively.)

You can also use this method to explicitly manage when your client receives input directed to the character such as when your application itself becomes active. For example, setting **State** to 2 makes the character topmost and your client receives all mouse and speech input events generated from user interaction with the character. Therefore, it also makes your client the input-active client of the character.

However, you can also set yourself to be the active client for a character without making the character topmost, by setting **State** to 1. This enables your client to receive input directed to that character when the character becomes topmost. Similarly, you can set your client to not be the active client (not to receive input) when the character becomes topmost, by setting **State** to 0.

Avoid calling this method directly after a **Show** method. **Show** automatically sets the input-active client. When the character is hidden, the **Activate** call may fail if it gets processed before the **Show** method completes.

If you call this method to a function, it returns a Boolean value that indicates whether the method succeeded. Attempting to call this method with the **State** parameter set to 2 when the specified character is hidden will fail. Similarly, if you set **State** to 0 and your application is the only client, this call fails because a character must always have a topmost client.

```
Dim Genie as Object

Sub FormLoad()

    Agent1.Characters.Load "Genie", _
        "C:\Program Files\Microsoft Agent\Characters\Genie.acs"

    Set Genie = Agent1.Characters ("Genie")

    If (Genie.Activate = True) Then
        'I'm active
    Else
        'I must be hidden or something
    End If

End Sub
```

## See Also

**ActivateInput** event, **DeactivateInput** event

---

## GestureAt Method

### Description

Plays the gesturing animation for the specified character at the specified location.

#### Syntax

*agent.Characters ("CharacterID").GestureAt X,Y*

Part	Description
<i>X,Y</i>	Required. An integer value that indicates the horizontal (X) screen coordinate and vertical (Y) screen coordinate to which the character will gesture. These coordinates must be specified in pixels.

#### Remarks

The server automatically plays the appropriate animation to gesture toward the specified location. The coordinates are always relative to the screen origin (upper left).

If you declare an object reference and set it to this method, it returns a **Request** object. In addition, if the associated animation has not been loaded on the local machine, the server sets the **Request** object's **Status** property to "failed" with an appropriate error number. Therefore, if you are using the HTTP protocol to access character animation data, use the **Get** method to load the **Gesturing** state animations before calling the **GestureAt** method.

-----

#### Get Method

##### Description

Retrieves specified animation data for the specified character.

##### Syntax

*agent.Characters ("CharacterID").Get Type, Name, [Queue]*

Part	Description
<i>Type</i>	Required. A string value that indicates the animation data type to load.  <b>"Animation"</b> A character's animation data. <b>"State"</b> A character's state data. <b>"WaveFile"</b> A character's audio (for spoken output) file.
<i>Name</i>	Required. A string that indicates the name of the animation type.  <b>"name"</b> The name of the animation or state.  For animations, the name is based on that defined for the character when saved using the Microsoft Agent Character Editor.  For states, the following values can be used:  <b>"Gesturing"</b> To get all <b>Gesturing</b> state animations. <b>"GesturingDown"</b> To get the <b>GesturingDown</b> animation.

	<p>"<b>GesturingLeft</b>" To get the <b>GesturingLeft</b> animation.</p> <p>"<b>GesturingRight</b>" To get the <b>GesturingRight</b> animation.</p> <p>"<b>GesturingUp</b>" To get the <b>GesturingUp</b> animation.</p> <p>"<b>Hiding</b>" To get the <b>Hiding</b> state animation.</p> <p>"<b>Hearing</b>" To get the <b>Hearing</b> state animation.</p> <p>"<b>Idling</b>" To get all <b>Idling</b> state animations.</p> <p>"<b>IdlingLevel1</b>" To get all <b>IdlingLevel1</b> animations.</p> <p>"<b>IdlingLevel2</b>" To get all <b>IdlingLevel2</b> animations.</p> <p>"<b>IdlingLevel3</b>" To get all <b>IdlingLevel3</b> animations.</p> <p>"<b>Listening</b>" To get the <b>Listening</b> state animation.</p> <p>"<b>Moving</b>" To get all <b>Moving</b> state animations.</p> <p>"<b>MovingDown</b>" To get the <b>MovingDown</b> animation.</p> <p>"<b>MovingLeft</b>" To get the <b>MovingLeft</b> animation.</p> <p>"<b>MovingRight</b>" To get the <b>MovingRight</b> animation.</p> <p>"<b>MovingUp</b>" To get the <b>MovingUp</b> animation.</p> <p>"<b>Showing</b>" To get the <b>Showing</b> state animation.</p> <p>"<b>Speaking</b>" To get the <b>Speaking</b> state animation.</p> <p>You can specify multiple animations and states by separating them with commas. However, you cannot mix types in the same <b>Get</b> statement.</p> <p>"<i>URL or filespec</i>" The specification for the sound (.WAV or .LWV) file. If the specification is not complete, it is interpreted as being relative to the specification used in the <b>Load</b> method.</p>
<i>Queue</i>	<p>Optional. A Boolean expression specifying whether the server queues the <b>Get</b> request.</p> <p><b>True</b> (Default) Queues the <b>Get</b> request. Any animation request that follows the <b>Get</b> request (for the same character) waits until the animation data is loaded.</p> <p><b>False</b> Does not queue the <b>Get</b> request.</p>

### Remarks

You need to use the **Get** method only to retrieve animation data using the HTTP protocol.

If you declare an object reference and set it to this method, it returns a **Request** object. If the associated animation fails to load, the server sets the **Request** object's **Status** property to "failed" with an appropriate error number. You can use the **RequestComplete** event to check the status and determine what action to take.

Animation or sound data retrieved with the **Get** method is stored in the browser's cache. Subsequent calls will check the cache, and if the animation data is already there, the control loads the data directly from the cache. Once loaded, the animation or sound data can be played with the **Play** or **Speak** methods.

### See Also

**Load** method



---

## Hide Method

### Description

Hides the specified character.

### Syntax

*agent*.Characters ("CharacterID").Hide [*Fast*]

Part	Description
<i>Fast</i>	Optional. A Boolean value that indicates whether to skip the animation associated with the character's Hiding state <b>True</b> Does not play the <b>Hiding</b> animation. <b>False</b> (Default) Plays the <b>Hiding</b> animation.

### Remarks

The server queues the actions of the **Hide** method in the character's queue, so you can use it to hide the character after a sequence of other animations. You can play the action immediately by using the **Stop** method before calling this method.

If you declare an object reference and set it to this method, it returns a **Request** object. In addition, if the associated **Hiding** animation has not been loaded and you have not specified the **Fast** parameter as **True**, the server sets the **Request** object **Status** property to "failed" with an appropriate error number. Therefore, if you are using the HTTP protocol to access character or animation data, use the **Get** method and specify the **Hiding** state to load the animation before calling the **Hide** method.

Hiding a character can also result in triggering the **ActivateInput** event of another client.

**Note** Hidden characters cannot access the audio channel. The server will pass back a failure status in the **RequestComplete** event if you generate an animation request and the character is hidden.

### See Also

**Show** method

---

## Interrupt Method

### Description

Interrupts the animation for the specified character.

### Syntax

*agent*.Characters ("CharacterID").Interrupt *Request*

## Remarks

You can use this to sync up animation between characters. For example, if another character is in a looping animation, this method will stop the loop and move to the next animation in the character's queue. You cannot interrupt a character animation that you are not using (that you have not loaded).

To specify the request parameter, you must create a variable and assign the animation request you want to interrupt:

```
Dim GenieRequest as Object
Dim RobbyRequest as Object
Dim Genie as Object
Dim Robby as Object

Sub FormLoad()

    MyAgent1.Characters.Load "Genie", _
        "C:\Program Files\Microsoft Agent\Characters\Genie.acs"

    MyAgent1.Characters.Load "Robby", _
        "C:\Program Files\Microsoft Agent\Characters\Robby.acs"

    Set Genie = MyAgent1.Characters ("Genie")
    Set Robby = MyAgent1.Characters ("Robby")

    Genie.Show

    Genie.Speak "Just a moment"

    Set GenieRequest = Genie.Play ("Processing")

    Robby.Show
    Robby.Play "confused"
    Robby.Speak "Hey, Genie. What are you doing?"
    Robby.Interrupt GenieRequest

    Genie.Speak "I was just checking on something."

End Sub
```

You cannot interrupt the animation of the same character you specify in this method because the server queues the **Interrupt** method in that character's animation queue. Therefore, you can only use **Interrupt** to halt the animation of another character you have loaded.

If you declare an object reference and set it to this method, it returns a **Request** object.

**Note**      **Interrupt** does not flush the character's queue; it halts the existing animation and moves on to the next animation in the character's queue. To halt and flush a character's queue, use the **Stop** method.

## See Also

**Stop** method

---

## MoveTo Method

### Description

Moves the specified character to the specified location.

### Syntax

*agent.Characters ("CharacterID").MoveTo X,Y, [Speed]*

Part	Description
<i>X,Y</i>	Required. An integer value that indicates the left edge (X) and top edge (Y) of the animation frame. Express these coordinates in pixels.
<i>Speed</i>	Optional. A Long integer value specifying in milliseconds how quickly the character's frame moves. The default value is 1000. Specifying zero (0) moves the frame without playing an animation.

### Remarks

The server automatically plays the appropriate animation assigned to the **Moving** states. The location of a character is based on the upper left corner of its frame.

If you declare an object variable and set it to this method, it returns a **Request** object. In addition, if the associated animation has not been loaded on the local machine, the server sets the **Request** object's **Status** property to "failed" with an appropriate error number. Therefore, if you are using the HTTP protocol to access character or animation data, use the **Get** method to load the **Moving** state animations before calling the **MoveTo** method.

Even if the animation is not loaded, the server still moves the frame.

**Note** If you call **MoveTo** with a non-zero value before the character is shown, it will return a failure status if you assigned it a **Request** object, because the non-zero value indicates that you are attempting to play an animation when the character is not visible.

-----

## Play Method

### Description

Plays the specified animation for the specified character.

### Syntax

*agent.Characters ("CharacterID").Play "AnimationName"*

Part	Description
------	-------------

<i>AnimationName</i>	Required. A string that specifies the name of an animation sequence.
----------------------	----------------------------------------------------------------------

### Remarks

An animation's name is defined when the character is compiled with the Microsoft Agent Character Editor. Before playing the specified animation, the server attempts to play the **Return** animation for the previous animation, if one has been assigned.

When accessing a character's animations using a conventional file protocol, you can simply use the **Play** method specifying the name of the animation. However, if you are using the HTTP protocol to access character animation data, use the **Get** method to load the animation before calling the **Play** method.

For more information, see the **Get** method.

To simplify your syntax, you can declare an object reference and set it to reference the **Character** object in the **Characters** collection and use the reference as part of your **Play** statements:

```
Dim Genie
Agent1.Characters.Load "Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf"

Set Genie = Agent1.Characters ("Genie")

Genie.Get "state", "Showing"
Genie.Show

Genie.Get "animation", "Greet, GreetReturn"
Genie.Play "Greet"
Genie.Speak "Hello."
```

If you declare an object reference and set it to this method, it returns a **Request** object. In addition, if you specify an animation that is not loaded or if the character has not been successfully loaded, the server sets the **Status** property of **Request** object to "failed" with an appropriate error number. However, if the animation does not exist and the character's data has already been successfully loaded, the server raises an error.

The **Play** method does not make the character visible. If the character is not visible, the server plays the animation invisibly, and sets the **Status** property of the **Request** object.

### Show Method

#### Description

Makes the specified character visible and plays its associated **Showing** animation.

#### Syntax

*agent*.Characters ("CharacterID").Show [*Fast*]

Part	Description
<i>Fast</i>	Optional. A Boolean expression specifying whether the server plays the <b>Showing</b>

	<p>animation.</p> <p><b>True</b> Skips the <b>Showing</b> state animation.</p> <p><b>False</b> (Default) Does not skip the <b>Showing</b> state animation.</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Remarks

If you declare an object reference and set it to this method, it returns a **Request** object. In addition, if the associated **Showing** animation has not been loaded and you have not specified the **Fast** parameter as **True**, the server sets the **Request** object's **Status** property to "failed" with an appropriate error number. Therefore, if you are using the HTTP protocol to access character animation data, use the **Get** method to load the **Showing** state animation before calling the **Show** method.

Avoid setting the **Fast** parameter to **True** without first playing an animation beforehand; otherwise, the character frame may display with no image. In particular, note that if you call **MoveTo** when the character is not visible, it does not play any animation. Therefore, if you call the **Show** method with **Fast** set to **True**, no image will display. Similarly, if you call **Hide** then **Show** with **Fast** set to **True**, there will be no visible image.

#### See Also

**Hide** method

#### Speak Method

##### Description

Speaks the specified text for the specified character.

##### Syntax

*agent*.**Characters** ("CharacterID").**Speak** [*Text*] [, *Url*]

Part	Description
<i>Text</i>	Optional. A string that specifies what the character says.
<i>Url</i>	Optional. A string expression specifying the specification for an audio file. The specification can be a file specification or URL.

#### Remarks

Although the *Text* and *Url* parameters are optional, one of them must be supplied. To use this method with a character configured to speak only in its word balloon or using a text-to-speech (TTS) engine, simply provide the *Text* parameter. Include a space between words to define appropriate word breaks in the word balloon, even for languages that do not traditionally include spaces.

You can also include vertical bar characters (|) in the *Text* parameter to designate alternative strings, so that the server randomly chooses a different string each time it processes the method.

Character support of TTS output is defined when the character is compiled using the Microsoft Agent Character Editor. To generate TTS output, a compatible TTS engine must already be installed before calling this method. For further information, see [Accessing Speech Support for Microsoft Agent](#).

If you use recorded sound-file output for the character, specify the file's location in the *Url* parameter. However, if you are using the HTTP protocol to access character or animation data, use the **Get** method to load the animation before calling the **Speak** method. When doing so, you still use the *Text* parameter to specify the words that appear in the character's word balloon. However, if you specify a linguistically enhanced sound file (.LWV) for the *Url* parameter and do not specify text for the word balloon, the *Text* parameter uses the text stored in the file.

You can also vary parameters of the speech output with special tags that you include in the *Text* parameter. For more information, see [Speech Output Tags](#). If you declare an object reference and set it to this method, it returns a **Request** object. In addition, if the file has not been loaded, the server sets the **Request** object's **Status** property to "failed" with an appropriate error code number.

The **Speak** method uses the last action played to determine which speaking animation to play. For example, if you preceded the **Speak** command with a **Play "GestureRight"**, the server will play **GestureRight** and then the **GestureRight** speaking animation.

If you call **Speak** and the audio channel is busy, the character's audio output will not be heard, but the text will display in the word balloon.

**Note** The word balloon's **Enabled** property must also be **True** for text to display.

**Note** If you are using a character that you did not compile, check the balloon **FontName** and **CharSet** properties for the character to determine whether they are appropriate for your locale. You may need to set these values before using the **Speak** method to ensure appropriate text display within the word balloon.

---

## Stop Method

### Description

Stops the animation for the specified character.

### Syntax

*agent.Characters ("CharacterID").Stop [Request]*

Part	Description
<i>Request</i>	Optional. To use this parameter, set the <b>Request</b> object in your code.

### Remarks

If you don't set the **Request** parameter, the server stops all animations for the character, including queued **Get** calls, and clears its animation queue unless the

character is currently playing its **Hiding** or **Showing** animation. This method does not stop non-queued **Get** calls.

To stop a specific animation or **Get** call, declare an object variable and assign your animation request to that variable:

```
Dim MyRequest
Dim Genie

Agent1.Characters.Load "Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf"

Set Genie = Agent1.Characters ("Genie")

Genie.Get "state", "Showing"
Genie.Get "animation", "Greet, GreetReturn"

Genie.Show

'This animation will never play
Set MyRequest = Genie.Play ("Greet")

Genie.Stop MyRequest
```

This method will not generate a **Request** object.

#### See Also

**StopAll** method

-----

### StopAll Method

#### Description

Stops all animation requests or specified types of requests for the specified character.

#### Syntax

*agent*.**Characters** ("CharacterID").**StopAll** [Type]

Part	Description
<i>Type</i>	Optional. To use this parameter you can use any of the following values. You can also specify multiple types by separating them with commas.  "Get" To stop all queued <b>Get</b> requests.  "NonQueuedGet" To stop all non-queued <b>Get</b> requests ( <b>Get</b> method with <b>Queue</b> parameter set to <b>False</b> ).  "Move" To stop all queued <b>MoveTo</b> requests.  "Play" To stop all queued <b>Play</b> requests.  "Speak" To stop all queued <b>Speak</b> requests.

#### Remarks

If you don't set the **Type** parameter, the server stops all animations for the character, including queued and non-queued **Get** requests, and clears its animation queue. It also stops playing a character's Hiding or Showing animation.

This method will not generate a **Request** object.

#### See Also

**Stop** method

---

#### Wait Method

##### Description

Causes the animation queue for the specified character to wait until the specified animation request completes.

##### Syntax

*agent*.**Characters** ("CharacterID").**Wait** Request

Part	Description
<i>Request</i>	A <b>Request</b> object specifying a particular animation. To set this parameter you must assign the <b>Request</b> object variable in your code.

#### Remarks

Use this method only when you support multiple (simultaneous) characters and are trying to sequence the interaction of characters (as a single client). (For a single character, each animation request is played sequentially--after the previous request completes.) If you have two characters and you want a character's animation request to wait until the other character's animation completes, set the **Wait** method to the other character's animation **Request** object, as shown in the following example:

```
Dim GenieRequest
Dim RobbyRequest
Dim Genie
Dim Robby

Sub window_Onload

Agent1.Characters.Load "Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf"
Agent1.Characters.Load "Robby", _
    "http://agent.microsoft.com/characters/robbby/robbby.acf"

Set Genie = Agent1.Characters("Genie")
Set Robby = Agent1.Characters("Robby")

Genie.Get "State", "Showing"
Robby.Get "State", "Showing"

Genie.Get "Animation", "Announce, AnnounceReturn, Pleased, _
    PleasedReturn"
```



```

Robby.Get "Animation", "Confused, ConfusedReturn, Sad, SadReturn"

Set Genie = Agent1.Characters ("Genie")
Set Robby = Agent1.Characters ("Robby")

Genie.MoveTo 100,100
Genie.Show

Robby.MoveTo 250,100
Robby.Show

Genie.Play "Announce"
Set GenieRequest = Genie.Speak ("Why did the chicken cross the road?")

Robby.Wait GenieRequest
Robby.Play "Confused"
Set RobbyRequest = Robby.Speak ("I don't know. Why did the chicken _
    cross the road?")

Genie.Wait RobbyRequest
Genie.Play "Pleased"
Set GenieRequest = Genie.Speak ("To get to the other side.")

Robby.Wait GenieRequest
Robby.Play "Sad"
Robby.Speak "I never should have asked."

End Sub

```

---

## Character Object Properties

The **Character** object exposes the following read-only properties:

**Description, ExtraData, HasOtherClients, Height, IdleOn, Left, MoveCause, Name, Pitch, SoundEffectsOn, Speed, Top, VisibilityCause, Visible, Width**

Note that the **Height, Left, Top, and Width** properties of a character differ from those that may be supported by the programming environment for the placement of the control. The **Character** properties apply to the visible presentation of a character, not the location of the Microsoft Agent control.

As with **Character** object methods, you can access a character's properties using the **Characters** collection, or simplify your syntax by declaring an object variable and setting it to a character in the collection. In the following example, Test1 and Test2 will be set to the same value:

```

Dim Genie
Dim MyRequest

Sub window_Onload

Agent.Characters.Load "Genie", _
    "http://agent.microsoft.com/characters/genie/genie.acf"

Set Genie = Agent.Characters("Genie")

```

```

Genie.MoveTo 15,15
MyRequest = Genie.Show()

End Sub

Sub Agent_RequestComplete (ByVal Request)

If Request = MyRequest Then
    Test1 = Agent.Characters("Genie").Top
    Test2 = Genie.Top
    MsgBox "Test 1 is " + cstr(Test1) + "and Test 2 is " + cstr(Test2)
End If

End Sub

```

Because the server loads a character asynchronously, ensure that the character has been loaded before querying its properties, for example, using the **RequestComplete** event. Otherwise, the properties may return incorrect values.

## Description Property

### Description

Returns or sets a string that specifies the description for the specified character.

### Syntax

*agent.Characters("CharacterID").Description*

### Remarks

The default value for the **Description** property for a character is defined when the character is compiled with the Microsoft Agent Character Editor.

**Note** The **Description** property setting is optional and may not be supplied for all characters.

---

## ExtraData Property

### Description

Returns a string that specifies additional data stored as part of the character.

### Syntax

*agent.Characters("CharacterID").ExtraData*

### Remarks

The default value for the **ExtraData** property for a character is defined when the character is compiled with the Microsoft Agent Character Editor. It cannot be changed or specified at run time.

**Note** The **ExtraData** property setting is optional and may not be supplied for all characters.

---

## HasOtherClients Property

### Description

Returns whether the specified character is in use by other applications.

### Syntax

*agent.Characters*("CharacterID").**HasOtherClients**

Value	Description
True	The character has other clients.
False	The character does not have other clients.

### Remarks

You can use this property to determine whether your application is the only or last client of the character, when more than one application is sharing (has loaded) the same character.

-----

## Height Property

### Description

Returns or sets the height of the specified character's frame.

### Syntax

*agent.Characters* ("CharacterID").**Height** [= *value*]

Part	Description
<i>value</i>	A Long integer that specifies the character's frame height.

### Remarks

The **Height** property is always expressed in pixels, relative to screen coordinates (upper left).

Even though the character appears in an irregularly shaped region window, the height of the character is based on the external dimensions of the rectangular animation frame used when the character was compiled with the Microsoft Agent Character Editor.

-----

## IdleOn Property

### Description

Returns or sets a Boolean value that determines whether the server manages the specified character's **Idling** state animations.

### Syntax

*agent.Characters ("CharacterID").IdleOn [=boolean]*

Part	Description
<b>True</b>	Server idle processing is enabled. The character's <b>Idling</b> animations are automatically played.
<b>False</b>	Server idle processing is disabled. The character's <b>Idling</b> animations are not automatically played.

### Remarks

The server automatically sets a time-out after the last animation played for a character. When this timer's interval is complete, the server begins the **Idling** state for a character, playing its associated **Idling** animations at regular intervals. The default value for the **IdleOn** property is **True**, meaning that the server manages the character's **Idling** state. If you want to manage the **Idling** state animations yourself, set the property to **False**.

-----

### Left Property

#### Description

Returns or sets the left edge of the specified character's frame.

#### Syntax

*agent.Characters ("CharacterID").Left [= value]*

Part	Description
<i>value</i>	A Long integer that specifies the left edge of the character's frame.

### Remarks

The **Left** property is always expressed in pixels, relative to screen origin (upper left).

Even though the character appears in an irregularly shaped region window, the location of the character is based on the external dimensions of the rectangular animation frame used when the character was compiled with the Microsoft Agent Character Editor.

-----

### MoveCause Property

#### Description

Returns an integer value that specifies what caused the character's last move.

### Syntax

*agent*.**Characters**("CharacterID").**MoveCause**

Value	Description
0	The character has not been moved.
1	The user moved the character.
2	Your application moved the character.
3	Another client application moved the character.

### Remarks

You can use this property to determine what caused the character to move, when more than one application is sharing (has loaded) the same character. These values are the same as those returned by the **Move** event.

### See Also

**Move** event, **MoveTo** method

-----

## Name Property

### Description

Returns or sets a string that specifies the default name of the specified character.

### Syntax

*agent*.**Characters** ("CharacterID").**Name**

### Remarks

The default value for the **Name** property for a character is defined when the character is compiled with the Microsoft Agent Character Editor. The server uses the **Name** property to automatically create commands for hiding and showing a character.

-----

## Pitch Property

### Description

Returns a Long integer for the specified character's current speech output (TTS) pitch setting.

### Syntax

*agent*.**Characters** ("CharacterID").**Pitch**

### Remarks

This property applies only to characters configured for TTS output. If the speech synthesis (TTS) engine is not enabled or installed, or the character does not support TTS output, this property returns zero (0).

Although your application cannot write this value, you can include **Pit** (pitch) tags in your output text that will temporarily increase the pitch for a particular utterance. For further information, see Speech Output Tags.

---

## SoundEffectsOn Property

### Description

Returns or sets whether sound effects are enabled for your character.

### Syntax

*agent*.**Characters**("CharacterID").**SoundEffectsOn** [=boolean]

Value	Description
<i>boolean</i>	A Boolean expression specifying whether sound effects are enabled. <b>True</b> Sound effects are enabled. <b>False</b> Sound effects are disabled.

### Remarks

This property determines whether sound effects included as a part of a character's animations will play when an animation plays.

### See Also

**SoundEffects** property

---

## Speed Property

### Description

Returns a Long integer that specifies the speed of the character's speech output.

### Syntax

*agent*.**Characters** ("CharacterID").**Speed**

### Remarks

This property returns the current speaking output speed setting for the character. For characters using TTS output, the property returns the actual TTS output for the character. If TTS is not enabled or the character does not support TTS output, the setting reflects the user setting for output speed.

Although your application cannot write this value, you can include **Spd** (speed) tags in your output text that will temporarily speed up the output for a particular utterance. For further information, see Speech Output Tags.

---

## Top Property

### Description

Returns or sets the top edge of the specified character's frame.

### Syntax

*agent*.**Characters** ("CharacterID").**Top** [= *value*]

Part	Description
<i>value</i>	A Long integer that specifies the character's top edge.

### Remarks

The **Top** property is always expressed in pixels, relative to screen origin (upper left).

Even though the character appears in an irregularly shaped region window, the location of the character is based on the external dimensions of the rectangular animation frame used when the character was compiled with the Microsoft Agent Character Editor.

Use the **MoveTo** method to change the character's location.

-----

## VisibilityCause Property

### Description

Returns an integer value that specifies what caused the character's visible state.

### Syntax

*agent*.**Characters**("CharacterID").**VisibilityCause**

Value	Description
0	The character has not been shown.
1	The user hid the character.
2	The user showed the character.
3	Your application hid the character.
4	Your application showed the character.
5	Another client application hid the character.
6	Another client application showed the character.

### Remarks

You can use this property to determine what caused the character to move when more than one application is sharing (has loaded) the same character. These values are the same as those returned by the **Show** and **Hide** events.

### See Also

**Hide** event, **Show** event, **Hide** method, **Show** method

-----

## Visible Property

### Description

Returns a Boolean indicating whether the character is visible.

### Syntax

*agent.Characters ("CharacterID").Visible*

Return	Description
<b>True</b>	The character is displayed.
<b>False</b>	The character is hidden (not visible).

### Remarks

To make a character visible or hidden, use the **Show** or **Hide** methods.

-----

## Width Property

### Description

Returns or sets the width of the frame for the specified character.

### Syntax

*agent.Characters ("CharacterID").Width [= value]*

Part	Description
<i>value</i>	A Long integer that specifies the character's frame width.

### Remarks

The **Width** property is always expressed in pixels.

Even though the character appears in an irregularly shaped region window, the location of the character is based on the external dimensions of the rectangular animation frame used when the character was compiled with the Microsoft Agent Character Editor.

-----

## The Commands Collection Object



The Microsoft Agent server maintains a list of commands that are currently available to the user. This list includes commands that the server defines for general interaction (such as Hide and Microsoft Agent Properties), the list of available (but non-input-active) clients, and the commands defined by the current active client. The first two sets of commands are global commands; that is, they are available at any time, regardless of the input-active client. Client-defined commands are available only when that client is input-active.

Each client application can define a collection of commands called the **Commands** collection. To add a command to the collection, use the **Add** or **Insert** method. Although you can specify a command's properties with separate statements, for optimum code performance, specify all of a command's properties in the **Add** or **Insert** method statement. For each command in the collection, you can determine whether user access to the command appears in the character's pop-up menu, in the Commands Window, in both, or in neither. For example, if you want a command to appear on the pop-up menu for the character, set the command's **Caption** and **Visible** properties. To display the command in the Commands Window, set the command's **Caption** and **Voice** properties.

A user can access the individual commands in your **Commands** collection only when your client application is input-active. Therefore, you'll typically want to set the **Caption** and **Voice** properties for the **Commands** collection object as well as for the commands in the collection, which places an entry for your **Commands** collection in a character's pop-up menu and in the Commands Window. When the user switches to your client by choosing its entry, the server automatically makes your client input-active and makes the commands in its collection available. This enables the server to present and accept only the commands that apply to the current input-active client's context. It also serves to avoid command-name collisions between clients.

When a character's pop-up menu displays, changes to the properties of a **Commands** collection or the commands in its collection do not appear until the user redisplay the menu. However, the Commands Window does display changes as they happen.

## Commands Object Methods

The server supports the following methods for the **Commands** collection object:

**Add, Command, Insert, Remove, RemoveAll**

### Add Method

#### Description

Adds a **Command** object to the **Commands** collection.

#### Syntax

*agent.Characters ("CharacterID").Commands.Add Name, Caption, Voice, \_  
Enabled, Visible*

Part	Description
<i>Name</i>	Required. A string value corresponding to the ID you assign for the command.
<i>Caption</i>	Optional. A string value corresponding to the name that will appear in the character's pop-up menu and in the Commands Window when the client application is input-active. For more information, see the Command object's

	<b>Caption</b> property.
<i>Voice</i>	Optional. A string value corresponding to the words or phrase to be used by the speech engine for recognizing this command. For more information on formatting alternatives for the string, see the <b>Command</b> object's <b>Voice</b> property.
<i>Enabled</i>	Optional. A Boolean value indicating whether the command is enabled. The default value is <b>True</b> . For more information, see the <b>Command</b> object's <b>Enabled</b> property.
<i>Visible</i>	Optional. A Boolean value indicating whether the command is visible in the character's pop-up menu for the character when the client application is input-active. The default value is <b>True</b> . For more information, see the <b>Command</b> object's <b>Visible</b> property.

#### Remarks

The value of a **Command** object's **Name** property must be unique within its **Commands** collection. You must remove a **Command** before you can create a new **Command** with the same **Name** property setting. Attempting to create a **Command** with a **Name** property that already exists raises an error.

#### See Also

**Insert** method, **Remove** method, **RemoveAll** method

---

### Command Method

#### Description

Returns a **Command** object in a **Commands** collection.

#### Syntax

*agent.Characters ("CharacterID").Commands.Command "Name"*

#### Remarks

You can use this method to access a **Command** object's properties.

**Note** This method may be required for some programming languages. It is not required for VBScript or Visual Basic. For further information on specifying **Command** methods, see Command Object Properties.

---

### Insert Method

#### Description

Inserts a **Command** object in the **Commands** collection.

#### Syntax

*agent.Characters ("CharacterID").Commands.Insert Name, RefName, Before, \_  
Caption, Voice, Enabled, Visible*

<b>Part</b>	<b>Description</b>
<i>Name</i>	Required. A string value corresponding to the ID you assign to the <b>Command</b> .
<i>RefName</i>	Required. A string value corresponding to the name (ID) of the command just above or below where you want to insert the new command.
<i>Before</i>	Optional. A Boolean value indicating whether to insert the new command before the command specified by RefName.  <b>True</b> (Default). The new command will be inserted before the referenced command.  <b>False</b> The new command will be inserted after the referenced command.
<i>Caption</i>	Optional. A string value corresponding to the name that will appear in the character's pop-up menu and in the Commands Window when the client application is input-active. For more information, see the Command object's <b>Caption</b> property.
<i>Voice</i>	Optional. A string value corresponding to the words or phrase to be used by the speech engine for recognizing this command. For more information on formatting alternatives for the string, see the <b>Command</b> object's <b>Voice</b> property.
<i>Enabled</i>	Optional. A Boolean value indicating whether the command is enabled. The default value is <b>True</b> . For more information, see the <b>Command</b> object's <b>Enabled</b> property.
<i>Visible</i>	Optional. A Boolean value indicating whether the command is visible in the Commands Window when the client application is input-active. The default value is <b>True</b> . For more information, see the <b>Command</b> object's <b>Visible</b> property.

#### Remarks

The value of a **Command** object's **Name** property must be unique within its **Commands** collection. You must remove a **Command** before you can create a new **Command** with the same **Name** property setting. Attempting to create a **Command** with a **Name** property that already exists raises an error.

#### See Also

Add method, Remove method, RemoveAll method

-----

#### Remove Method

##### Description

Removes a **Command** object from the **Commands** collection.

##### Syntax

*agent.Characters ("CharacterID").Commands.Remove Name*

<b>Part</b>	<b>Description</b>
<i>Name</i>	Required. A string value corresponding to the ID for the command.

**Remarks**

When a **Command** object is removed from the collection, it no longer appears when the character's pop-up menu is displayed or in the Commands Window when your client application is input-active.

**See Also**

**RemoveAll** method

-----

**RemoveAll Method****Description**

Removes all **Command** objects from the **Commands** collection.

**Syntax**

*agent*.**Characters** ("CharacterID").**Commands.RemoveAll**

**Remarks**

When a **Command** object is removed from the collection, it no longer appears when the character's pop-up menu is displayed or in the Commands Window when your client application is input-active.

**See Also**

**Remove** method

-----

**Commands Object Properties**

The server supports the following properties for the **Commands** collection:

**Caption, Count, Visible, Voice**

An entry for the **Commands** collection can appear in both the pop-up menu and the Commands Window for a character. To make this entry appear, set its **Caption** property. The following table summarizes how the properties of a **Commands** object affect the entry's presentation:

<u>Caption Property</u>	<u>Voice Property</u>	<u>Visible Property</u>	<u>Appears in Character's Pop-up Menu</u>	<u>Appears in Commands Window</u>
Yes	Yes	True	Yes	Yes
Yes	No	True	Yes	No
Yes	Yes	False	No	Yes
Yes	No	False	No	No
No	Yes	True	No	No*

<u>Caption Property</u>	<u>Voice Property</u>	<u>Visible Property</u>	<u>Appears in Character's Pop-up Menu</u>	<u>Appears in Commands Window</u>
No	Yes	False	No	No*
No	No	True	No	No
No	No	False	No	No

\*The command is still voice-accessible. If the client is input-active and has commands in its collection, "(command undefined)" appears in the Commands Window.

## Caption Property

### Description

Determines the text displayed for the **Commands** object in the character's pop-up menu and in the Commands Window.

### Syntax

*agent.Characters ("CharacterID").Commands.Caption [=string]*

<u>Part</u>	<u>Description</u>
<i>string</i>	A string expression that evaluates to the text displayed as the caption.

### Remarks

If you define commands for a **Commands** collection that have their **Caption**, **Enabled**, and **Voice** properties set, you would typically also define **Caption** and **Voice** settings for the associated **Commands** collection. If the **Commands** collection has no **Voice** or no **Caption** setting and is currently input-active, but the commands in its collection have **Caption** and **Voice** settings, the commands appear in the Commands Window tree view under "(undefined command)" when your client application becomes input-active.

-----

## Count Property

### Description

Returns a Long integer (read-only property) that specifies the count of **Command** objects in the **Commands** collection.

### Syntax

*agent.Characters ("CharacterID").Commands.Count*

### Remarks

**Count** includes only the number of **Command** objects you define in your **Commands** collection. Server or other client entries are not included.

-----

## Visible Property

### Description

Returns or sets a value that determines whether your **Commands** collection's caption appears in the character's pop-up menu.

### Syntax

*agent.Characters ("CharacterID").Commands.Visible* [= *boolean*]

Part	Description
<i>boolean</i>	A Boolean expression specifying whether your <b>Commands</b> object appears in the character's pop-up menu.  <b>True</b> The <b>Caption</b> for your <b>Commands</b> collection appears.  <b>False</b> The <b>Caption</b> for your <b>Commands</b> collection does not appear.

### Remarks

This property must be set to **True** for commands in your collection to appear in the pop-up menu when your application is input-active.

-----

## Voice Property

### Description

Returns or sets the text that is passed to the speech engine (for recognition).

### Syntax

*agent.Characters ("CharacterID").Commands.Voice* [= *string*]

Part	Description
<i>string</i>	A string value corresponding to the words or phrase to be used by the speech engine for recognizing this command.

### Remarks

If you do not supply this parameter, the caption for your **Commands** object will not appear in the Commands Window.

The string expression you supply can include square bracket characters ([ ]) to indicate optional words and vertical bar characters, (|) to indicate alternative strings. Alternates must be enclosed in parentheses. For example, "(hello [there] | hi)" tells the speech engine to accept "hello," "hello there," or "hi" for the command. Remember to include appropriate spaces between the text that's in brackets or parentheses and the text that's not in brackets or parentheses.

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

When defining the word grammar for your command, always make sure that you include at least one word that is required; that is, avoid supplying only optional words. In addition, make sure that the word includes only pronounceable words and letters. For numbers, it is better to spell out the word than use the numeric representation. Also, omit any punctuation or symbols. For example, instead of "the #1 \$10 pizza!", use "the number one ten dollar pizza". Including non-pronounceable characters or symbols for one command may cause the speech engine to fail to compile the grammar for all your commands. Finally, make your voice parameter as distinct as reasonably possible from other voice commands you define. The greater the similarity between the voice grammar for commands, the more likely the speech engine will make a recognition error. You can also use the confidence scores to better distinguish between two commands that may have similar or similar-sounding voice grammar.

**Note** The operation of this property depends on the state of the server's speech recognition property. For example, if speech recognition is disabled or not installed, this parameter has no effect. If speech recognition is enabled during a session, however, the command will become accessible when its client application is input-active.

-----

## The Command Object

A **Command** object is an item in a **Commands** collection. The server provides the user access to your **Command** objects when your client application becomes input-active.

To access the property of a **Command** object, you reference it in its collection using its **Name** property. In VBScript and Visual Basic you can use the **Name** property directly:

```
agent.Characters("CharacterID").Commands("Name").property [= value]
```

For programming languages that don't support collections, use the **Command** method:

```
agent.Characters("CharacterID").Commands.Command("Name").property [= value]
```

You can also reference a Command object by creating a reference to it. In Visual Basic, declare an object variable and use the Set statement to create the reference:

```
Dim Cmd1 as Object
...
Set Cmd1 = Agent.Characters("MyCharacterID").Commands("SampleCommand")
...
Cmd1.Enabled = True
```

In Visual Basic 5.0, you can also declare the object as type **IAgentCtlCommand** and create the reference. This convention enables early binding, which results in better performance:

```

Dim Cmd1 as IAgentCtlCommand
...
Set Cmd1 = Agent.Characters("MyCharacterID").Commands("SampleCommand")
...
Cmd1.Enabled = True

```

In VBScript, you can declare a reference as a particular type, but you can still declare the variable and set it to the **Command** in the collection:

```

Dim Cmd1
...
Set Cmd1 = Agent.Characters("MyCharacterID").Commands("SampleCommand")
...
Cmd1.Enabled = True

```

A command may appear in either the character's pop-up menu and the Commands Window, or in both. To appear in the pop-up menu it must have a caption and have the **Visible** property set to **True**. In addition, its Commands collection object **Visible** property must also be set to **True**. To appear in the Commands Window, a **Command** must have its **Caption** and **Voice** properties set. Note that a character's pop-up menu entries do not change while the menu displays. If you add or remove commands or change their properties while the character's pop-up menu is displayed, the menu displays those changes whenever the user next displays it. However, the Commands Window dynamically reflects any changes you make.

The following table summarizes how the properties of a **Command** affect its presentation:

<u>Caption Property</u>	<u>Voice Property</u>	<u>Visible Property</u>	<u>Enabled Property</u>	<u>Appears in Character's Pop-up Menu</u>	<u>Appears in Commands Window</u>
Yes	Yes	True	True	Normal	Yes
Yes	Yes	True	False	Disabled	No
Yes	Yes	False	True	Does not appear	Yes
Yes	Yes	False	False	Does not appear	No
Yes	No	True	True	Normal	No
Yes	No	True	False	Disabled	No
Yes	No	False	True	Does not appear	No
Yes	No	False	False	Does not appear	No
No	Yes	True	True	Does not appear	No*



<u>Caption Property</u>	<u>Voice Property</u>	<u>Visible Property</u>	<u>Enabled Property</u>	<u>Appears in Character's Pop-up Menu</u>	<u>Appears in Commands Window</u>
No	Yes	True	False	Does not appear	No
No	Yes	False	True	Does not appear	No*
No	Yes	False	False	Does not appear	No
No	No	True	True	Does not appear	No
No	No	True	False	Does not appear	No
No	No	False	True	Does not appear	No
No	No	False	False	Does not appear	No

\*The command is still voice-accessible.

Generally, if you define commands with **Voice** settings, you also define **Caption** and **Voice** settings for its associated **Commands** collection. If a **Commands** collection has no **Voice** or no **Caption** setting and is currently input-active, but its **Command** objects do have **Caption** and **Voice** settings and their **Enabled** properties are **True**, the **Command** objects appear in the Commands Window tree view under "(undefined command)" when your client application becomes input-active.

When the server receives input for one of your commands, it sends a **Command** event, and passes back the name of the **Command** as an attribute of the **UserInput** object. You can then use conditional statements to match and process the **Command**.

## Command Object Properties

The following **Command** properties are supported:

**Caption, Confidence, ConfidenceText, Enabled, Visible, Voice**

## Caption Property

### Description

Determines the text displayed for a **Command** in the specified character's pop-up menu and the Commands Window.

### Syntax

*agent.Characters ("CharacterID").Commands("name").Caption [= string]*

Part	Description
<i>string</i>	A string expression that evaluates to the text displayed as the caption for the <b>Command</b> .

-----

## Confidence Property

### Description

Returns or sets whether the client's **ConfidenceText** appears in the Listening Tip.

### Syntax

*agent.Characters ("CharacterID").Commands("name").Confidence [= number]*

Part	Description
<i>number</i>	A numeric expression that evaluates to a Long integer that identifies confidence value for the <b>Command</b> .

### Remarks

If the returned confidence value of the best match (**UserInput.Confidence**) does not exceed value you set for the **Confidence** property, the text supplied in **ConfidenceText** is displayed in the Listening Tip.

-----

## ConfidenceText Property

### Description

Returns or sets the client's **ConfidenceText** that appears in the Listening Tip.

### Syntax

*agent.Characters ("CharacterID").Commands("name").ConfidenceText [= string]*

Part	Description
<i>string</i>	A string expression that evaluates to the text for the <b>ConfidenceText</b> for the <b>Command</b> .

### Remarks

When the returned confidence value of the best match (**UserInput.Confidence**) does not exceed the **Confidence** setting, the server displays the text supplied in **ConfidenceText** in the Listening Tip.

-----

### Enabled Property

#### Description

Returns or sets whether the **Command** is enabled in the specified character's pop-up menu.

#### Syntax

*agent.Characters ("CharacterID").Commands("name").Enabled [= boolean]*

Part	Description
<i>boolean</i>	A Boolean expression specifying whether the <b>Command</b> is enabled. <b>True</b> The <b>Command</b> is enabled. <b>False</b> The <b>Command</b> is disabled.

### Remarks

If the **Enabled** property is set to **True**, the **Command** object's caption appears as normal text in the character's pop-up menu when the client application is input-active. If the **Enabled** property is **False**, the caption appears as unavailable (disabled) text. A disabled **Command** is also not accessible for voice input.

-----

### Visible Property

#### Description

Returns or sets whether the **Command** is visible in the character's pop-up menu.

#### Syntax

*agent.Characters ("CharacterID").Commands("name").Visible [= boolean]*

Part	Description
<i>boolean</i>	A Boolean expression specifying whether the <b>Command</b> 's caption appears in the character's pop-up menu. <b>True</b> (Default) The caption appears. <b>False</b> The caption does not appear.

### Remarks

Set this property to **False** when you want to include voice input for your own interfaces without having them appear in the pop-up menu for the character. If you set a **Command** object's **Caption** property to the empty string (""), the caption text will not appear in the pop-up menu (for example, as a blank line), regardless of its **Visible** property setting.

The **Visible** property setting of a **Command** object's parent **Commands** collection does not affect the **Visible** property setting of the **Command**.

-----

### Voice Property

#### Description

Returns or sets the text that is passed to the speech engine grammar (for recognition) for matching this **Command** for the character.

#### Syntax

*agent.Characters ("CharacterID").Commands ("name").Voice [= string]*

Part	Description
<i>string</i>	A string value corresponding to the words or phrase to be used by the speech engine for recognizing this <b>Command</b> .

### Remarks

If you do not supply this parameter, the caption for your **Commands** object will not appear in the Commands Window. If you specify a **Voice** parameter but not a **Caption**, the command will not appear in the Commands Window, but it will be voice-accessible when the client application becomes input-active.

Your string expression can include square bracket characters ([ ]) to indicate optional words and vertical bar characters (|) to indicate alternative strings. Alternates must be enclosed in parentheses. For example, "(hello [there] | hi)" tells the speech engine to accept "hello," "hello there," or "hi" for the command. Remember to include appropriate spaces between the text that's in brackets or parentheses and the text that's not in brackets or parentheses.

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

When defining the word grammar for your command, always make sure that you include at least one word that is required; that is, avoid supplying only optional words. In

addition, make sure that the word includes only pronounceable words and letters. For numbers, it is better to spell out the word rather than using the numeric representation. Also, omit any punctuation or symbols. For example, instead of "the #1 \$10 pizza!", use "the number one ten dollar pizza". Including non-pronounceable characters or symbols for one command may cause the speech engine to fail to compile the grammar for all your commands. Finally, make your voice parameter as distinct as reasonably possible from other voice commands you define. The greater the similarity between the voice grammar for commands, the more likely the speech engine will make a recognition error. You can also use the confidence scores to better distinguish between two commands that may have similar or similar-sounding voice grammar.

**Note** The operation of this property depends on the state of the server's speech recognition property. For example, if speech recognition is disabled or not installed, this property has no effect.

---

## The Balloon Object

Microsoft Agent supports textual captioning of spoken output using a cartoon word balloon. A character's initial word balloon window defaults are defined and compiled in the Microsoft Agent Character Editor. Once running, the word balloon's **Enabled** and **Font** properties may be overridden by the user. If a user changes the word balloon's properties, they affect all characters. You can access the properties for a character's word balloon through the **Balloon** object, which is a child of the **Characters** collection.

The **Balloon** object supports the following properties:

**BackColor**, **BorderColor**, **CharSet**, **CharsPerLine**, **Enabled**, **FontName**, **FontBold**, **FontItalic**, **FontSize**, **FontStrikeThru**, **FontUnderline**, **ForeColor**, **NumberOfLines**, **Visible**

## BackColor Property

### Description

Returns the background color currently displayed in the word balloon window for the specified character.

### Syntax

*agent.Characters ("CharacterID").Balloon.BackColor*

### Remarks

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

---

## BorderColor Property

### Description

Returns the border color currently displayed for the word balloon window for the specified character.

**Syntax**

*agent.Characters ("CharacterID").Balloon.BorderColor*

**Remarks**

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

-----

**CharSet Property**

**Description**

Returns or sets the character set used for the font displayed in the specified character's word balloon.

**Syntax**

*agent.Characters ("CharacterID").Balloon.CharSet [= value]*

Part	Description																
<i>value</i>	<p>A integer value that specifies the character set used by the font. The following are some common settings for value:</p> <table><tr><td>0</td><td>Standard Windows® characters (ANSI).</td></tr><tr><td>1</td><td>Default character set.</td></tr><tr><td>2</td><td>The symbol character set.</td></tr><tr><td>128</td><td>Double-byte character set (DBCS) unique to the Japanese version of Windows.</td></tr><tr><td>129</td><td>Double-byte character set (DBCS) unique to the Korean version of Windows.</td></tr><tr><td>134</td><td>Double-byte character set (DBCS) unique to the Simplified Chinese version of Windows.</td></tr><tr><td>136</td><td>Double-byte character set (DBCS) unique to the Traditional Chinese version of Windows.</td></tr><tr><td>255</td><td>Extended characters normally displayed by DOS applications.</td></tr></table> <p>For other character set values, consult the Microsoft Win32® documentation.</p>	0	Standard Windows® characters (ANSI).	1	Default character set.	2	The symbol character set.	128	Double-byte character set (DBCS) unique to the Japanese version of Windows.	129	Double-byte character set (DBCS) unique to the Korean version of Windows.	134	Double-byte character set (DBCS) unique to the Simplified Chinese version of Windows.	136	Double-byte character set (DBCS) unique to the Traditional Chinese version of Windows.	255	Extended characters normally displayed by DOS applications.
0	Standard Windows® characters (ANSI).																
1	Default character set.																
2	The symbol character set.																
128	Double-byte character set (DBCS) unique to the Japanese version of Windows.																
129	Double-byte character set (DBCS) unique to the Korean version of Windows.																
134	Double-byte character set (DBCS) unique to the Simplified Chinese version of Windows.																
136	Double-byte character set (DBCS) unique to the Traditional Chinese version of Windows.																
255	Extended characters normally displayed by DOS applications.																

**Remarks**

The default value for the character set of a character's word balloon is set in the Microsoft Agent Character Editor. In addition, the user can override the character-set settings for all characters in the Microsoft Agent property sheet.

**Note** If you are using a character that you did not compile, check the **FontName** and **CharSet** properties for the character to determine whether they are appropriate for your locale. You may need to set these values before using the **Speak** method to ensure appropriate text display within the word balloon.

#### See Also

**FontName** property

---

### CharsPerLine Property

#### Description

Returns the characters per line supported for the word balloon for the specified character.

#### Syntax

*agent.Characters ("CharacterID").Balloon.CharsPerLine*

#### Remarks

The **CharsPerLine** property returns the average number of characters (letters) being displayed in the word balloon as a Long integer value.

---

### Enabled Property

#### Description

Returns whether the word balloon is enabled for the specified character.

#### Syntax

*agent.Characters ("CharacterID").Balloon.Enabled*

#### Remarks

The **Enabled** property returns a Boolean value specifying whether the balloon is enabled by the user. **True** indicates the **Balloon** is enabled. **False** indicates it is not enabled (displayed).

The word balloon can also be disabled as part of a character's definition when the character is compiled in the Microsoft Agent Character Editor. If a character is defined to not support the word balloon, this property will always be **False** for the character.

---

### FontName Property

#### Description

Returns or sets the font used in the word balloon for the specified character.

#### Syntax

*agent.Characters ("CharacterID").Balloon.FontName [ = font]*

Part	Description
<i>font</i>	A string value corresponding to the font's name.

#### Remarks

The **FontName** property defines the font used to display text in the word balloon window as a string. The default value for the font settings of a character's word balloon are set in the Microsoft Agent Character Editor. In addition, the user can override font settings for all characters in the Microsoft Agent property sheet.

**Note** If you are using a character that you did not compile, check the **FontName** and **CharSet** properties for the character to determine whether they are appropriate for your locale. You may need to set these values before using the **Speak** method to ensure appropriate text display within the word balloon.

#### See Also

**CharSet** property

-----

### FontBold Property

#### Description

Returns the font style currently displayed in the word balloon window for the specified character.

#### Syntax

*agent.Characters ("CharacterID").Balloon.FontBold*

#### Remarks

The **FontBold** property returns a Boolean value specifying whether the font is bold. **True** indicates the font is bold. **False** indicates the font is not bold.

The default value for the font settings of a character's word balloon are set in the Microsoft Agent Character Editor. In addition, the user can override font settings for all characters in the Microsoft Agent property sheet.

-----

### FontItalic Property

#### Description

Returns the font style currently displayed in the word balloon window for the specified character.

#### Syntax

*agent.Characters ("CharacterID").Balloon.FontItalic*



#### Remarks

The **FontItalic** property returns a Boolean value specifying whether the font is italic. **True** indicates the font is italic. **False** indicates the font is not italic.

The default value for the font settings of a character's word balloon are set in the Microsoft Agent Character Editor. In addition, the user can override font settings for all characters in the Microsoft Agent property sheet.

---

### FontSize Property

#### Description

Returns or sets the font size supported for the word balloon for the specified character.

#### Syntax

*agent.Characters ("CharacterID").Balloon.FontSize* [= *points*]

Part	Description
<i>points</i>	A Long integer value specifying the font size in points.

#### Remarks

The **FontSize** property returns a Long integer value specifying the current font size in points. The maximum value for **FontSize** is 2160 points.

The default value for the font settings of a character's word balloon are set in the Microsoft Agent Character Editor. In addition, the user can override font settings for all characters in the Microsoft Agent property sheet.

---

### FontStrikeThru Property

#### Description

Returns the font style currently displayed in the word balloon window for the specified character.

#### Syntax

*agent.Characters ("CharacterID").Balloon.FontStrikeThru*

#### Remarks

The **FontStrikeThru** property returns a Boolean value specifying whether the font uses the strikethrough effect. **True** indicates the font uses the strikethrough effect. **False** indicates the font does not use the strikethrough effect.

The default value for the font settings of a character's word balloon are set in the Microsoft Agent Character Editor. In addition, the user can override font settings for all characters in the Microsoft Agent property sheet.

---

## FontUnderline Property

### Description

Returns the font style currently displayed in the word balloon window for the specified character.

### Syntax

*agent.Characters ("CharacterID").Balloon.FontUnderline*

### Remarks

The **FontUnderline** property returns a Boolean value specifying whether the font is underlined. **True** indicates the font is underlined. **False** indicates the font is not underlined.

The default value for the font settings of a character's word balloon are set in the Microsoft Agent Character Editor. In addition, the user can override font settings for all characters in the Microsoft Agent property sheet.

---

## ForeColor Property

### Description

Returns the foreground color currently displayed in the word balloon window for the specified character.

### Syntax

*agent.Characters ("CharacterID").Balloon.ForeColor*

### Remarks

The **ForeColor** property returns a value that specifies the color of text in the word balloon. The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

---

## NumberOfLines Property

### Description

Returns the number of lines supported for the word balloon for the specified character.

### Syntax

*agent.Characters ("CharacterID").Balloon.NumberOfLines*

### Remarks

The **NumberOfLines** property returns the number of lines of text as a Long integer value.

---

## Visible Property

### Description

Returns or sets the visible setting for the word balloon for the specified character.

### Syntax

*agent*.**Characters** ("CharacterID").**Balloon.Visible** [=boolean]

Part	Description
<i>boolean</i>	A Boolean expression specifying whether the word balloon is visible. <b>True</b> The balloon is visible. <b>False</b> The balloon is hidden.

### Remarks

If you attempt to set this property while the character is speaking, moving, or being dragged, the property setting does not take effect until the preceding operation is completed. Calling the **Speak** method automatically makes the balloon visible, setting the **Visible** property to **True**. If the character's balloon AutoHide property is enabled, the balloon is automatically hidden after the output text is spoken. Clicking or dragging a character that is not currently speaking also automatically hides the balloon even if its AutoHide property is disabled. (A character's word-balloon AutoHide property can only be set in the Microsoft Agent Character Editor. The property is not exposed in the API.)

---

## The AudioOutput Object

The **AudioOutput** object provides access to audio output properties maintained by the server. The properties are read-only, but the user can change them in the Microsoft Agent property sheet.

### Enabled, SoundEffects

## Enabled Property

### Description

Returns a Boolean indicating whether audio (spoken) output is enabled.

### Syntax

*agent*.**AudioOutput.Enabled**

### Remarks

When the **Enabled** property returns **True**, the **Speak** method produces audio output. When it returns **False**, it means that speech output is not installed or has been disabled by the user. Only the user can set this property value.

---

## SoundEffects Property

### Description

Returns a Boolean indicating whether sound effects (.WAV) files configured as part of a character's actions will play.

### Syntax

*agent*.AudioOutput.SoundEffects

### Remarks

When the **SoundEffects** property returns **True**, sound effects included in a character's definition will be played. When **False**, the sound effects will not be played. Only the user can set this property value.

## The SpeechInput Object

The **SpeechInput** object provides access to the speech input properties maintained by the server. The properties are read-only for client applications, but the user can change them in the Microsoft Agent property sheet. The server returns values only if a compatible speech engine has been installed and is enabled.

## SpeechInput Properties

If a speech recognition engine is installed and enabled, accessing these properties will start the speech engine:

**Enabled, Engine, HotKey, Installed, Language, ListeningTip**

## Enabled Property

### Description

Returns a Boolean value indicating whether speech input is enabled.

### Syntax

*agent*.SpeechInput.Enabled

### Remarks

The **Enabled** property reflects the state of the speech recognition engine set in the **Engine** property. Values for other **SpeechInput** properties do not change when **Enabled** is set to **False**. Querying this property raises an error if no speech engine has been installed.

**Note** The user can override this property.

---

## Engine Property

### Description

Returns or sets the speech recognition engine that is currently selected for input.

### Syntax

*agent.SpeechInput.Engine* [=modeID]

Part	Description
<i>modeID</i>	A string expression specifying the mode ID of the speech engine.

### Remarks

The **Engine** property takes a string value specifying the mode ID of the speech engine. Querying this property raises an error if **Installed** or **Enabled** is **False**.

**Note** The user can override this property.

---

## HotKey Property

### Description

Returns a string that specifies the user's current setting for the push-to-talk hot key.

### Syntax

*agent.SpeechInput.HotKey*

### Remarks

Querying this property raises an error if **Installed** or **Enabled** is **False**.

---

## Installed Property

### Description

Returns a Boolean that indicates whether a compatible speech engine is installed.

### Syntax

*agent.SpeechInput.Installed*

### Remarks

If no compatible speech engine has been installed, this property returns **False**. However, querying any other **SpeechInput** properties raises an error. Therefore, check

this property before checking the values of the **SpeechInput** or **CommandsWindow** objects.

---

## Language Property

### Description

Returns a string that specifies what language is configured for speech input.

### Syntax

*agent.SpeechInput.Language*

### Remarks

This property value is set based on the selected speech recognition engine set in the **Engine** property. Querying this property raises an error if **Installed** or **Enabled** is **False**.

---

## ListeningTip Property

### Description

Returns a Boolean indicating whether the server displays the Listening Tip.

### Syntax

*agent.SpeechInput.ListeningTip*

### Remarks

When **ListeningTip** returns **True**, the server displays the tip window when the user presses the push-to-talk hot key. Querying this property raises an error if **Installed** or **Enabled** is **False**.

## The CommandsWindow Object

The **CommandsWindow** object provides access to properties of the Commands Window. The Commands Window is a shared resource primarily designed to enable users to view voice-enabled commands. If speech recognition is disabled, the Commands Window is also disabled, but you can still read its property settings. If no speech engine is installed or if speech is disabled, querying for the **CommandsWindow** properties raises an error.

## CommandsWindow Properties

**Height, Left, Top, Visible, Width**

## Height Property

### Description

Returns an integer value specifying the current height, in pixels, of the Commands Window.

**Syntax**

*agent.CommandsWindow.Height*

**Remarks**

The server displays the Commands Window based on the position and size set by the user. Querying this property raises an error if no speech engine has been installed.

-----

**Left Property**

**Description**

Returns an integer value specifying the left edge, in pixels, of the Commands Window.

**Syntax**

*agent.CommandsWindow.Left*

**Remarks**

The server displays the Commands Window based on the position and size set by the user. Querying this property raises an error if no speech engine has been installed.

-----

**Top Property**

**Description**

Returns an integer value specifying the top edge, in pixels, of the Commands Window.

**Syntax**

*agent.CommandsWindow.Top*

**Remarks**

The server displays the Commands Window based on the position and size set by the user. Querying this property raises an error if no speech engine has been installed.

-----

**Visible Property**

**Description**

Returns or sets whether the Commands Window is visible (open).

**Syntax**

*agent.CommandsWindow.Visible* [=boolean]

Part	Description
------	-------------

<i>boolean</i>	A Boolean expression specifying whether the Commands Window is visible. <b>True</b> The window is visible. <b>False</b> The window is hidden (closed).
----------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Remarks

The server displays the window based on the position and size set by the user. Querying this property raises an error if no speech engine has been installed.

**Note** The user can override this property.

-----

### Width Property

#### Description

Returns an integer value specifying the width, in pixels, of the Commands Window.

#### Syntax

*agent.CommandsWindow.Width*

#### Remarks

The server displays the Commands Window based on the position and size set by the user. Querying this property raises an error if no speech engine has been installed.

### The PropertySheet Object

The **PropertySheet** object provides several properties you can use if you want to manipulate the character relative to the Microsoft Agent property sheet:

#### Height, Left, Page, Top, Visible, Width

If you query **Height**, **Left**, **Top**, and **Width** properties before the property sheet has ever been shown, their values return as zero (0). Once shown, these properties return the last position and size of the window (relative to your current screen resolution).

### Height Property

#### Description

Returns an integer value specifying the current height, in pixels, of the Microsoft Agent property sheet window.

#### Syntax

*agent.PropertySheet.Height*



**Remarks**

The server displays the window based on the location set by the user.

-----

**Left Property****Description**

Returns an integer value specifying the current left edge, in pixels, of the Microsoft Agent property sheet window.

**Syntax**

*agent*.**PropertySheet.Left**

**Remarks**

The server displays the window based on the location set by the user.

-----

**Page Property****Description**

Returns or sets the page displayed in the Microsoft Agent property sheet window.

**Syntax**

*agent*.**PropertySheet.Page** [= *string*]

Part	Description
<i>string</i>	A string expression with one of the following values. <b>"Speech"</b> Displays the Speech Recognition page. <b>"Output"</b> Displays the Output page. <b>"Copyright"</b> Displays the Copyright page.

**Remarks**

If no speech engine is installed, setting **Page** to **"Speech"** has no effect. Also, the window's **Visible** property must be set to **True** for the user to see the page.

**Note** The user can override this property.

-----

**Top Property****Description**

Returns an integer value specifying the current top edge, in pixels, of the Microsoft Agent property sheet window.

**Syntax**

*agent*.**PropertySheet.Top**

**Remarks**

The server displays the window based on the location set by the user.

-----

**Visible Property**

**Description**

Returns or sets whether the Microsoft Agent property sheet window is visible (open).

**Syntax**

*agent*.**PropertySheet.Visible** [=*boolean*]

<b>Part</b>	<b>Description</b>
<i>boolean</i>	A Boolean expression specifying whether the window is visible. <b>True</b> The window is visible. <b>False</b> The window is hidden (closed).

**Remarks**

The server displays the window based on the location and size set by the user.

**Note** The user can override this property.

-----

**Width Property**

**Description**

Returns an integer value specifying the current width, in pixels, of the Microsoft Agent property sheet window.

**Syntax**

*agent*.**PropertySheet.Width**

**Remarks**

The server displays the window based on the location set by the user.

