

# Microsoft Agent Programming Interface Overview

---

## ActiveX™ Technology for Interactive Software Agents



August 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

Licensing and Distribution

Animation Services

Input Services

Output Services

## Introduction

The Microsoft Agent API provides services that support the display and animation of animated characters. Implemented as an OLE Automation (Component Object Model [COM]) server, Microsoft Agent enables multiple applications, called *clients* or *client applications*, to host and

access its animation, input, and output services at the same time. A client can be any application that supports the Microsoft Agent's COM interfaces.

Although you can call Microsoft Agent's COM interfaces directly, Microsoft Agent also includes an ActiveX™ control. This control makes it easy to access Microsoft Agent's services from programming languages that support the ActiveX control interface. For information, see *Programming the Microsoft Agent Server Interface* and *Programming the Microsoft Agent Control*.

As a COM server, Microsoft Agent starts up only when a client application requests to connect to it. It remains running until all clients close their connection. When no connected clients remain, Microsoft Agent automatically exits. Microsoft Agent also exits when a user explicitly chooses the Exit command on the pop-up menu of Microsoft Agent's taskbar icon and confirms exiting in the warning message box. This action causes the server to send a **Shutdown** event to all connected clients advising them that the server is exiting.

## Licensing and Distribution

The Microsoft Agent self-extracting executable installs a number of system files and registry entries. Web developers can include the CLSID in the <OBJECT> tag of their HTML page, subject to the provisions of the license agreement displayed when the control is downloaded and installed. Application developers who want to add Microsoft Agent services and any of its components (including Microsoft Agent character files) to their application must obtain a redistribution license for Microsoft Agent. For more information on redistribution of Microsoft Agent, see Microsoft Agent Licensing and Redistribution.

## Animation Services

Microsoft Agent's animation services manage the animation and movement of a character's image in its own window on the screen. An animation is defined as a sequence of timed and optionally branched frames, composed of one or more images. Specifying the **Play** statement with the name of an animation plays that animation. Animation names are specific to a character definition. As an animation plays, the shape of its window changes to match the image in the frame. This results in a movable graphic image, or *sprite*, displayed on top of the desktop and all windows.

Each client application can display and animate its own character. You can also share a character between multiple client applications. Microsoft Agent also supports clients using multiple characters displayed at the same time. The animation services enable you to animate characters independently or synchronize their animation.

To access a character, use the **Load** method to load the character's data. Microsoft Agent's services include a data provider that supports two formats for loading character and animation data: a single structured file and separate files. Typically, you would use the single file format (.ACS) when the data can be stored locally. The multiple file format (.ACF,.AAF) works best when you want to download animations individually, such as when accessing animations from a Web page script.

Microsoft Agent provides a set of characters you can download and use, subject to the provisions of the license agreement. For information on accessing the characters, see the Microsoft Agent Characters page.

You can define your own character and its animations using any rendering tool you prefer. To compile a character's animations for use with Microsoft Agent, use the Microsoft Agent Character Editor. This tool enables you to define a character's default properties as well as define animations for the character. The Microsoft Agent Character Editor also enables you to select the appropriate file format when you create a character. For alternative formats or rendering, you can supply your own animation data provider.

The animation services also play certain animations automatically. For example, when you call the **MoveTo** and **GestureAt** methods, the server determines what animation to play based on the character's current position. Similarly, the services play **Idling** animations when the user has not interacted with the character. These server-managed animations are called "states," and are defined when a character is created. For more information, see Using The Microsoft Agent Character Editor.

Client applications can directly hide or show a character by using the **Hide** or **Show** methods that play the animations assigned to the **Hiding** and **Showing** states and set the character's **Visible** property. This functionality enables you to display or hide a character using your own interface.

Although the server produces no output when a character is hidden, the server still queues and processes the animation request (plays the animation), but passes a request status back to the client. In the hidden state, the character cannot become input-active. However, if the user speaks the name of a hidden character (when speech input is enabled), the server automatically shows the character.

Microsoft Agent queues animation requests and processes them asynchronously. This enables your application's code to continue while character animations play. However, you can still monitor and manage your character's animation queue by creating an object reference to the request.

## Input Services

A client application provides the primary user interface for interaction with a character. You can program a character to respond to any form of input, from button-clicks to typed-in text. In addition, Microsoft Agent provides events so you can program what happens when the user clicks, double-clicks, or drags the character. The server passes the coordinates of the pointer and any modifier key state for these events.

## Speech Input Support

In addition to supporting mouse and keyboard interaction, Microsoft Agent includes support for speech input. You can use The Microsoft Command and Control Engine for supporting speech recognition. The Command and Control speech engine enables users to speak naturally without pausing between words. Speech recognition is speaker-independent, but it can be trained for improved performance. Because Microsoft Agent's support for speech input is based on Microsoft SAPI (Speech Application Programming Interface), you can use Microsoft Agent with other engines that are SAPI-compliant.

The user can initiate speech input by pressing and holding the push-to-talk listening hotkey. In this mode, if the speech engine receives the beginning of spoken input, it holds the audio channel open until it detects the end of the utterance. However, when not receiving input, it does not block audio output. This enables the user to issue multiple voice commands while holding down the key, and the character can respond when the user isn't speaking. If a character

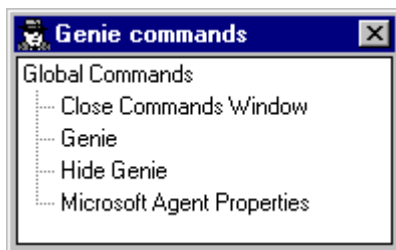
attempts to speak while the user is speaking, the character's audible output fails though text may still be displayed in its word balloon. If the character has the audio channel while listening key is pressed, the server automatically transfers control back to the user after processing the text in the **Speak** method. An optional MIDI tone is played to cue the user to begin speaking. This enables the user to provide input even if the application driving the character failed to provide logical pauses in its output.

Because multiple client applications can share the same character and because multiple clients can use different characters at the same time, the server designates one client as the *input-active* client and sends mouse and voice input only to that client application. This maintains the orderly management of user input, so that an appropriate client responds to the input. Typically, user interaction determines which client application becomes input-active. For example, if the user clicks a character, that character's client application becomes input-active. Similarly, if a user speaks the name of a character, it becomes input-active. Also, when the server processes a character's **Show** method, the client of that character becomes input-active. In addition, you can call the **Activate** method to make your client input-active, but you should do so only when your client application is active. For example, if the user clicks your application's window, activating your application, you can call the **Activate** method to receive and process mouse and speech input.

If multiple clients use the same character, the server defines the last one shown or the last one input-active as the current input-active character. However, you can also use the **Activate** method to set your client to become input-active or remain non-input-active when the user selects that character.

## The Commands Window

When a speech engine is installed and enabled, Microsoft Agent also includes a special interface called the Commands Window. This window displays the voice-enabled commands defined by the client applications for a character.



**Figure 1. Commands Window**

The Commands Window appears when the Open Commands Window command is chosen. Client commands appear in the Commands Window based on the **Caption** and **Voice** property settings of their **Commands** collection object.

The server creates a set of voice commands for general interaction and displays these under the Global Commands entry.

Caption	Voice Grammar
Open   Close Commands Window	((open   show) [the] commands [window]   what can I say [now])  toggles with:  close [the] commands [window]
<i>CharacterName</i>	[show] <i>CharacterName</i> *
Hide All Characters	hide all [characters]
Hide <i>CharacterName</i>	hide <i>CharacterName</i> **
Microsoft Agent Properties   Close Microsoft Agent Property Sheet	[(open   show)] [microsoft] agent (properties   property sheet)  toggles with:  close [microsoft] agent (properties   property sheet)

\* All loaded characters are listed.

\*\* A character is listed here only if it is currently visible.

The server automatically displays the commands of the current input-active client and, if necessary, scrolls the window to display as many of the client's commands as possible, based on the size of the window. If the character has no client entries, the Global Commands entry is expanded. Non-input-active clients appear in the tree as single entries.

Speaking the voice command for a client's **Commands** collection switches to that client and the Commands Window displays the commands of that client. No other entries are expanded. Similarly, if the user switches characters, the Commands Window changes to display the commands of its input-active client. If the client is already input-active, speaking its voice command has no effect. (However, if the user collapses the active client's subtree with the mouse, speaking the client name redisplay the client's subtree.) If a client has voice commands, but no **Voice** setting for its **Commands** object (or no **Caption**), the tree displays "(command undefined)" as the parent entry, but only when that client is input-active and the client has commands in its collection that have **Caption** and **Voice** settings. The server includes voice commands in the Global Commands entry ([show] [me] global commands). If the user speaks "Global Commands," the Commands Window always displays its associated subtree entries. If they are already displayed, the command has no effect.

Although you can also display or hide the Commands Window from your application's code using the **Visible** property, you cannot change the Commands Window size or location. The server maintains the Commands Window's properties based on the user's interaction with the window. Its initial location is immediately adjacent to the Microsoft Agent taskbar icon.

The Commands Window is included in the ALT+TAB window order. This enables a user to switch to the window to scroll, resize, or reposition the window with the keyboard.

## The Listening Tip

When speech input is installed, Microsoft Agent includes a special tooltip window that displays when the user presses the push-to-talk hot key. The following table summarizes the display of the Listening Tip when speech recognition is enabled.

Action	Result
User presses the listening mode hot key	<p>The Listening Tip appears below the active client's character and displays:</p> <p style="text-align: center;">-- <i>CharacterName</i> is listening -- for "<i>InputActiveClientCaption</i>" commands.</p> <p>The first line identifying the character is centered. The second line is left justified and breaks to a third line when it exceeds the Listening Tip's maximum width.</p> <p>If an input-active client of the character does not have a caption or defined voice parameters for its <b>Commands</b> object, the Listening Tip displays:</p> <p style="text-align: center;">-- <i>CharacterName</i> is listening -- for commands.</p> <p>If there are no visible characters, the Listening Tip appears adjacent to the Microsoft Agent taskbar icon and displays:</p> <p style="text-align: center;">-- All characters are hidden -- Say the name of a character to display it.</p> <p>If the speech recognition is still initializing, the Listening Tip displays:</p> <p style="text-align: center;">Say the name of a character to display it.</p> <p>If the audio channel is busy, as when the character is audibly speaking or some other application is using the audio channel, the Listening Tip displays:</p> <p style="text-align: center;">-- <i>CharacterName</i> is preparing to listen -- Please wait to speak.</p>
User presses the listening mode hot key and speaks a voice command	<p>The Listening Tip appears below the active client's character and displays:</p> <p style="text-align: center;">-- <i>CharacterName</i> heard -- "<i>CommandText</i>"</p> <p>The first line is centered. The second line is left justified and breaks to a third line when it exceeds the Listening Tip's maximum width.</p>

The Listening Tip automatically times out after being presented. If the "heard" text time-out completes when the user presses the hot key, the tip reverts to the "listening" text unless the server receives another matching utterance. In this case, the tip displays the new "heard" text and begins the time-out for that tip text. If the user releases the hot key and the server is displaying the "heard" text, the time-out continues. However, although the server displays the "listening" text, it immediately removes the Listening Tip when the user releases the hot key.

The Listening Tip does not appear when the pointer is over the Microsoft Agent taskbar icon. Instead, the standard notification tip window appears and displays "Press the *name of hot key* key to talk to *InputActiveCharacterName*" when the server is enabled. If all characters are hidden, the tip displays, "Press the *name of hot key* key and say the name of a character." However, if the user presses the speaking hot key, the tip reflects the same text as the listening tip. For example, it displays, "*CharacterName* is listening for *InputActiveClientCaption* commands," or, "*CharacterName* is listening for commands," if the input-active client has not defined its **Caption** property; and "*CharacterName* heard *CommandText*," when the speech engine processes a recognition. When the user disables speech input (or speech recognition is

not installed), the icon's tooltip displays, "Microsoft Agent is running." When the server is in its suspended state, the tip displays, "Microsoft Agent is suspended."

Clients cannot write directly to the Listening Tip, but you can specify alternative "heard" text that the server displays on recognition of a matching voice command. To do this, set the **Confidence** property and the new **ConfidenceText** property for the command. If spoken input matches the command, but the best match does not exceed the confidence setting, the server uses the text set in the **ConfidenceText** property in the tip window. If the client does not supply this value, the server displays the text (grammar) it matched.

## Pop-up Menu Support

Microsoft Agent includes a pop-up menu (also known as a contextual menu) for each character that the server displays automatically when a user right-clicks the character. This menu displays some standard commands managed by the server, but it also enables you to add and remove commands that your client application defines. The current input-active client's commands appear, provided that their **Caption** and **Visible** properties have been set. If the **Enabled** property has been set to **True**, the command appears enabled; if **False**, the command appears disabled (unavailable appearance). You define the access key for the entry by including an ampersand (&) before the text character of the **Caption** text setting. A separator appears before these commands. To create entries on a character's pop-up menu, define a **Commands** collection object and set the **Caption** and **Visible** properties of the commands. Note that menu entries do not change while the menu displays. If you add or remove commands or change their properties, the menu displays the changes when the user redisplay the menu.

The captions of any other clients (non-input-active) appear after another separator. To appear in the list, the **Caption** and **Visible** properties of their associated **Commands** object must be set. An ampersand in the text setting of the **Caption** property defines the access key for the entry. It is possible that access keys for menu items may be non-unique; however, this cannot be avoided. Separators appear only when there are items in the menu to separate. If no entries exist, the separator for that group does not appear.

Because the server provides the right-click pop-up menu as a standard service, avoid defining your own pop-up menu on the right-click event. However, if you define your own character, you can disable the server's pop-up menu by using the Microsoft Agent Character Editor. This enables you to support your own interface for the right-click action for your character. However, the pop-up menu cannot be disabled by a client application.

When the user selects a command from a character's pop-up menu or the Commands Window, the server triggers the **Command** event of the associated client and passes back the parameters of the input using the **UserInput** object.

The server also provides a pop-up menu for the Microsoft Agent taskbar icon. This menu provides the user access to all connected characters, and automatically assigns access keys for the characters based on the first letter of the character name. The menu also includes an entry that provides user access to the Microsoft Agent property sheet. You cannot modify the contents of the Microsoft Agent taskbar pop-up menu.

When the user chooses Exit from the Microsoft Agent taskbar icon pop-up menu, the server notifies the user that applications (clients with existing connections to the server) may not operate correctly and requests confirmation. If the user confirms shutting down the server, the server sends all client applications a **Shutdown** event. Your application becomes responsible for how it handles this state. Client applications cannot stop or cancel server shutdown.

If the server gets a request to restart after being shut down, for example, because a new client connects, the server partially reloads in suspended state, displaying the "suspended" Microsoft Agent icon in the taskbar notification area. Microsoft Agent also displays a message box indicating that the current application has requested to restart its services and offers to restart the server. (It also includes an option for the user not to be prompted again.) If the user chooses to restart, the server restores in its full operation and sends clients the **Restart** event. If the user chooses not to restart, the server remains its suspended state until all its clients close their connections or the user explicitly chooses to exit it again.

## The Microsoft Agent Property Sheet

The Microsoft Agent property sheet provides options for users to adjust their interaction with all characters. For example, users can disable speech input or change input parameters. Users can also change the output settings for the word balloon. These settings override any that are set by a client application or set as part of the character definition. Your application cannot change or disable these options, because they apply to the general user preferences for operation of all characters. However, your application can display or close the property sheet and access the location of its window.

## Output Services

Microsoft Agent also supports audio output for the character. This includes spoken output and sound effects. For spoken output, the server lip-syncs the character's defined mouth images to the output. You can choose text-to-speech (TTS) synthesis, recorded audio, or only word balloon text output.

## Synthesized Speech Support

If you use synthesized speech, your character has the ability to say almost anything, which provides the greatest flexibility. With recorded audio, you can give the character a specific or unique voice. To specify output, provide the spoken text as a parameter of the **Speak** method.

Because Microsoft Agent's architecture uses Microsoft SAPI for synthesized speech output, you should be able to use any engine that conforms to this specification, and support International Phonetic Alphabet (IPA) output using the **Visual** method of the **ITTSNotifySinkW** interface. For further information on the SAPI interfaces, see the Microsoft Speech SDK.

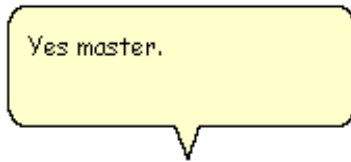
## Audio Output Support

Microsoft Agent enables you to use audio files for a character's spoken output. You can record audio files and use the **Speak** method to play that data. Microsoft Agent animation services automatically support lip-syncing the character mouth by using the audio characteristics of the audio file. Microsoft Agent also supports a special format for audio files, which includes additional phoneme and word-break information for more enhanced lip-sync support. You can generate this special format using the Microsoft Linguistic Information Sound Editing Tool (coming soon).

## Word Balloon Support



Spoken output can also appear as textual output in the form of a cartoon word balloon. This can be used to supplement the spoken output of a character or as an alternative to audio output.



**Figure 2. The Word Balloon**

Word balloons support only captioned communication from the character, *not* user input. Therefore, the word balloon does not support input controls. If you want to provide user input for a character, supply those interfaces from your application or the other input services provided by Microsoft Agent, such as the pop-up menu.

When you define a character, you can specify whether to include word balloon support. However, if you use a character that includes word balloon support, you cannot disable the support.

### **Animation Sound Effects**

Microsoft Agent also enables you to include sound effects as a part of a character's animation. Using the Microsoft Agent Character Editor, you can specify the filename of standard Windows® sound (.WAV) files to trigger on a given frame. Note that Microsoft Agent does not mix sound effects and spoken output, so spoken output does not begin until a sound effect completes. Therefore, avoid any long or looping sound effect as a part of a character's animation.