



AW Scripting Language (alphabetical listing)

NOTE: The scripting language is used only with ActiveWords PLUS!. ActiveWords SE users considering an [upgrade](#) to ActiveWords PLUS! may find this information helpful.

ActiveWord scripting commands below are used in the Action field of the Scripting editor in the [Add Wizard](#) included with ActiveWords PLUS!. Use these commands by themselves or in combination to make your ActiveWords do what you want.

<ABOUT ACTIVEWORDS>
Displays ActiveWords About dialog, a help file with information about an item selected in a Directory.
<ACTIVEWORDS MENU>
Opens the ActiveWords menu. Example: <code><ACTIVEWORDS MENU>o</code> will select the <i>Options...</i> option from the menu.
<ACTIVEWORDS OPTIONS>
Opens the dialog used to set ActiveWords options.
<ADD TEMPORAL ACTIVEWORD>
Adds the last line typed by the user as a temporal ActiveWord, using the last word as the Code. You can add up to 30 Temporal ActiveWords. Example: <code>This is an example mycode addtmp</code> where <code>addtmp</code> is an ActiveWord with the <code><ADD TEMPORAL ACTIVEWORD></code> command, will add <code>This is an example</code> as a Temporal ActiveWord assigned to the <code>mycode</code> ActiveWord.
<ADD WIZARD> <ADD WIZARD:PROGRAM> <ADD WIZARD:DOCUMENT> <ADD WIZARD:INTERNET> <ADD WIZARD:FOLDER> <ADD WIZARD:SETTINGS> <ADD WIZARD:SUBSTITUTION> <ADD WIZARD:MAILTO> <ADD WIZARD:COMMAND> <ADD WIZARD:Any Agent Type>
Opens the Add Wizard . You can skip the Wizard's main dialog by specifying the type of ActiveWord you want to add (i.e. Program, Document, Folder, Internet Site or Windows Setting). If the type specified isn't any of the standard, it is looked up among the registered Agents to pass control to its respective owner.
<ALT> </ALT>
<code><ALT></code> simulates pressing the Alt key, <code></ALT></code> simulates releasing the Alt key. An <code><ALT></code> must always be closed by an <code></ALT></code> .

Example:

<ALT>f</ALT> will open the File menu.

<ALTGR> </ALTGR>

Same as Alt but with the AltGr key. This is the Right Alt key and is different from the left key on some keyboards for special characters.

Example:

On a Spanish Traditional keyboard layout, <ALTGR>2</ALTGR> will type the @ sign.

<ALWAYS FLOAT>

Prevents the Monitor bar from being able to park (dock) on any side of the screen if the user drags the bar there.

<ASCII:###>

Types the character represented by the ASCII value (specified through ###). Same as doing an Alt+### combination from the keyboard.

Example:

<ASCII:0153> will type the ™ symbol.

<AUTO HIDE>

When the Monitor bar is docked, it will hide when it doesn't have the mouse pointer over it.

<AWCURRENTPROFILE>

<AW CURRENT PROFILE>

The name of the ActiveWords Profile currently being used is returned. [Pre-processed](#)

Example:

I'm currently using profile <AW CURRENT PROFILE>.

<AWCURRENTUSER>

<AW CURRENT USER>

The name of the ActiveWords Current User is returned. [Pre-processed](#)

Example:

The current user is <AW CURRENT USER>.

<AWCURRENTVERSION>

<AW CURRENT VERSION>

The Version and Build number of the ActiveWords you are currently using. [Pre-processed](#)

Example:

The following Bug Report is for ActiveWords version <AW CURRENT VERSION>.

<AW CURRENT WINDOW CLASS>

Identifies the Class Name of the window where the current ActiveWord is being executed. [Pre-processed](#)

ActiveWords normally sends Shift+Backspace keys to remove a triggered ActiveWord from the context in which it was entered. This scripting command may be used to identify the Window Class Name of those contexts where Shift+Backspace keys do not remove a triggered ActiveWord. After identifying the Window Class Name an exception may be created in the AWSettings.ini file that will send keys other than Shift+Backspace when a Window with the identified Class Name is the active context.

Example:

<WORK PAD>Window Class Name = "<AW CURRENT WINDOW CLASS>".

Trigger this Scripting Command in the context of the program where you wish to identify its Window Class Name. The command will pop-up a Work Pad window with the identified Window Class Name.

Tip: With the Window to identify in focus, launch the AW ActionPad and trigger the ActiveWord for this command. This will avoid any undesired effect of typing the ActiveWord over the Window that you wish to identify.

You may then use this identified Window Class Name to setup exceptions to the way ActiveWords erases triggered words on this program. To enter an exception:

- Shutdown ActiveWords.
- Make a backup copy of the file AWSettings.ini located in the ActiveWords installation directory (usually "C:\Program Files\ActiveWords").
- Edit AWSettings.ini in a text editor (like Notepad) and use the Class Name identified by this command to add an exception of one of the following types:

To send Backspace keys (instead of the default Shift+Backspace), add an exception like this:

[Backspace]
Backspace#="Window Class Name"

To send a single Escape key, add an exception like this:

[Escape]
Escape#="Window Class Name"

To send no keys (not erase triggered ActiveWords), add an exception like this:

[Nothing]
Nothing#="Window Class Name"

Please note that these sections (denoted by square brackets) may already exist in the AWSettings.ini file. Add a new exception to the existing section replacing # with the next consecutive number, starting from 0 (zero).

<AWDIR>
<ACTIVEWORDSDIR>
<ACTIVE WORDS DIR>

The complete path where ActiveWords is installed is returned (i.e. "C:\Program Files\ActiveWords"). [Pre-processed](#)

Example:

To open the AWDevelopmentCenter, the Navigation action field would be as follows:

"<AWDIR>\AWDevelopmentCenter.exe"

<AW FILE:path>

The path points to a text file which contents will be analyzed as a script, resulting in either a plain substitution or a command.

Example:

<AW FILE:d:\awscript.txt> will read the file awscript.txt from drive d: and execute the text as if it were a standard ActiveWords script.

<AW PASTE FILE:path>

The path points to a text file whose contents will be sent as a text substitution through the clipboard.

Example:

`<AW PASTE FILE:d:\document.txt>` will read the file document.txt from drive d: and paste its text content to the current application.

<BACK SPACE[:##]>

<BACKSPACE[:##]>

<BS[:##]>

Simulates the Backspace key. The optional parameter in brackets specifies the ## of times to press the Backspace key.

Example:

`<BACKSPACE>` will simulate pressing the Backspace key once.

`<BACKSPACE:10>` will simulate pressing the Backspace key 10 times.

<CHECK LAST AW>

Brings up an Editor for the last ActiveWord that was triggered.

<CLEAR TEMPORAL ACTIVEWORDS>

Clears all temporal ActiveWords.

<CLOSE AWMONITOR>

Exit ActiveWords by closing the AWMonitor bar.

<CLOSE DOCUMENT>

Close the current document. Works only for MDI (Multiple Document Interface) Applications like MSWord. Same as pressing Ctrl+W

<CLOSE WINDOW>

Close the current window. Please note that this command will not close the entire application if the current window is not the main window. Same as Alt+F4

<CODE[:Stop | :Continue]>

Where *CODE* is the ActiveWord of any item in the current WordBase where the Confirmation Flag is unchecked.

Example: An item exists for Microsoft Word with the ActiveWord 'msw', where the Confirmation Flag has been unchecked.

Action: `<msw><ctrl>o</ctrl>`; will launch Microsoft Word and present the File Open dialog.

The *CODE* must be different from any ActiveWords scripting language command. You can use chains of *CODES* in a Script and each nested *CODE*'s script can also include embedded *CODES*. You can use as many chained *CODES* as you want in a script. You can also use up to ten levels of recursion, where *CODE1* calls *CODE2*, *CODE2* calls *CODE3* ... *CODE9* calls *CODE10*.

Example:

Consider a group of 5 users in a computer network. The administrator has a *CODE* to send a message to each one of the users, 'send1' for user 1, 'send2' for user 2 and so on. The following script would send the message to all 5 users:

ActiveWord: `SendAll`

Action: `<send1><send2><send3><send4><send5>`

You can combine the use of embedded ActiveWords with the conditional commands `<Only>`, `<Not>` and `<InputBox>` by using the :Stop and :Continue parameters in the following way:

If the :Stop parameter is specified, the rest of the Script that follows the embedded ActiveWord won't be executed upon successful completion of the embedded ActiveWord's

script. That is, if the embedded ActiveWord's script has the command `<Only:App1>`, the rest of the calling Script will stop if the current application is App1; if the embedded ActiveWord's script has the command `<Not:App1>`, the rest of the calling Script will stop if the current application is not App1; if the embedded ActiveWord's script has the command `<InputBox>`, the rest of the calling Script will stop if the user presses Ok on the InputBox.

If the `:Continue` parameter is specified, the rest of the Script that follows the embedded ActiveWord will be executed upon successful completion of the embedded ActiveWord's script. That is, if the embedded ActiveWord's script has the command `<Only:App1>`, the rest of the calling Script will continue if the current application is App1; if the embedded ActiveWord's script has the command `<Not:App1>`, the rest of the calling Script will continue if the current application is not App1; if the embedded ActiveWord's script has the command `<InputBox>`, the rest of the calling Script will continue if the user presses Ok on the InputBox.

Example:

Consider the following ActiveWords:

ActiveWord 1 Code: NotepadFind

Action: `<ONLY:Notepad.exe><ALT>s</ALT>f`

ActiveWord 2 Code: ExplorerFind

Action: `<ONLY:Explorer.exe><WINSTART>f</WINSTART>`

ActiveWord 3 Code: Find

Action: `<NotepadFind:Stop><ExplorerFind:Stop><CTRL>f</CTRL>`

Note that ActiveWord number 3 uses ActiveWords 1 and 2 embedded with the `:Stop` parameter. Also, both ActiveWords 1 and 2 use the command `<Only:...>` in their respective scripts. This gives us the following possible scenarios:

- If you run ActiveWord 3 on Notepad.exe, the embedded *ActiveWord* NotepadFind will be successfully executed because the `<ONLY:Notepad.exe>` command will return TRUE. The rest of the Script in ActiveWord 3 (ExplorerFind and Ctrl+F) will not be executed because NotepadFind has the `:Stop` parameter.
- If you run ActiveWord 3 on any application but Notepad.exe or Explorer.exe, ActiveWord 1 and ActiveWord 2 won't be executed (because their corresponding `<Only:...>` clause will return FALSE) and the fragment `<CTRL>f</CTRL>` from ActiveWord 3 will be executed.

This example demonstrates a Generic Find ActiveWord.

**`<CTRL> </CTRL>`
`<CONTROL> </CONTROL>`**

`<CTRL>` simulates pressing the Ctrl key, `</CTRL>` simulates releasing the Ctrl key. A `<CTRL>` must always be closed by a `</CTRL>`.

Example:

`<CTRL>v</CTRL>` will paste the contents of the clipboard.

**`<DEL[: ##]>`
`<DELETE[: ##]>`**

Simulates the Delete key. The optional parameter in brackets specifies the ## of times to press the Delete key.

Example:

`<DELETE:5>` will simulate pressing the delete key 5 times.

`<DELAY[: ####]>`

Waits 700 milliseconds (.7 seconds) [or waits the number of milliseconds indicated by the

number, up to 10,000 (10 seconds)].

Example:

`<DELAY:5000>` will wait 5 seconds before continuing.

<DIRECTORY:name[:ADD WIZARD[=Editor]][:RESTORE={YES|NO}][:OnOffSwitch] [:ShowAllSwitch][:ShowFindTools][:PREFIX="any text "][:SUFFIX="any text "]>

Opens the Directory *name* containing all ActiveWords assigned to that Directory. Note that the Directory *name* need not be the same as the ActiveWord that opens the Directory.

If the optional ADD WIZARD parameter is used, the Directory will show an Add icon on its toolbar to add ActiveWords to that Directory. The ADD WIZARD parameters follows the same syntax as the [<ADD WIZARD>](#) command. The PREFIX or SUFFIX parameters are used to add a personalized prefix or suffix to the description of the ActiveWords that will be added through the ADD WIZARD command.

The optional RESTORE parameter specifies whether the Directory should remain visible when adding an ActiveWord (Yes) or hidden (No). Default is Yes.

Specifying the ShowAllSwitch displays a button on the Directory that allows a user to show the Visible/Not-visible column to hide or show specific items.

Specifying the OnOffSwitch displays a button on the Directory which allows the user to turn On/Off specific items.

Specifying the ShowFindTools will display a search box allowing the user to narrow the Directory results to items containing a certain string. Only items in the current Directory are searched.

Examples:

`<DIRECTORY:AWControls>` will open a Directory with all the AWControls.

`<DIRECTORY:Sites:ADD WIZARD=INTERNET:RESTORE=NO>` will open a Directory with all the ActiveWords in the Sites Directory, with an Add button on the toolbar which when clicked will open the Internet Wizard. Any ActiveWords added will automatically belong to the Sites Directory. The Directory will close when the Add button is used.

`<DIRECTORY:Programs: OnOffSwitch: ShowAllSwitch:Add Wizard=command :Prefix="This ActiveWord will open the ":Suffix= program.>`

<DLL:DllName.dll:Function [:Ignore]>

Calls the specified function from a DLL, passing the current line as a parameter, and types the return value from the call. The "Function" parameter is case sensitive and when used with the Ignore parameter, the return value from the Dll call is ignored (nothing is typed as a result). Since it is [Pre-processed](#), the `<delay>` command won't take effect, unless the Dll call is used embedded within another Script.

This command is intended to be used mainly by AW Agents who want to extend the capability of ActiveWords, by calling a library function. Examples of these agents are the AW NumberToText Agent and the AW Math Agent.

Example:

`<DLL:AWDateTime\AWDateTime.dll:ReturnDate>` will load the AWDateTime DLL from the `<AWDIR>\AWApps\AWDateTime` folder, run its ReturnDate function and type out the result of the call (today's Date).

<DOCK ON BOTTOM>

Park the Monitor bar on the lower side of the screen.
<DOCK ON TOP>
Park the Monitor bar on the upper side of the screen.
<DOWN[:##]>
<p>Simulates the Down arrow key. The optional parameter in brackets specifies the ## of times to press the Down arrow key.</p> <p>Example: <code><DOWN:25></code> will simulate pressing the down key 25 times.</p>
<END>
<p>Simulates the End key (goes to end of line or bottom of a list).</p> <p>Example: <code><END></code> in a word processor will put the cursor at the end of the current line. <code><CTRL><END></CTRL></code> will go to the end of a document.</p>
<ENTER[:##]>
<RETURN[:##]>
<p>Simulates the Enter key. The optional parameter in brackets specifies the ## of times to press the Enter key.</p> <p>Example: <code><ENTER></code> on a "Save As..." dialog will Save and close the dialog. <code><ENTER:2></code> in a word processor will insert 2 new lines.</p>
<ERASE LAST WORD>
<DELETE LAST WORD>
Deletes the last word typed.
<ESC>
<p>Simulates the Escape key.</p> <p>Example: <code><ESC></code> on a "Save As..." dialog will Cancel and close the dialog.</p>
<EVENT> </EVENT[:\TimeOut]>
<p>The Event command will get the handle of the foremost Window. The /Event part of the command will wait until this Window is not the foremost Window anymore. The /Event command will wait for the number of seconds specified in the TimeOut parameter. The default is 5 seconds if nothing is specified.</p> <p>Example: <code><Event><WinStart>f</WinStart></Event:10> Notepad<Enter></code> will get the foremost window (where this command is executed), invoke the Windows Find Files dialog and wait (up to 10 seconds) until the dialog appears (which will be the new foremost Window). Then it will type Notepad and press the Enter key to begin searching.</p>
<F#>
<p>Simulates the Function Keys F1 through Fn, where n is the highest numeric Function Key on your keyboard.</p> <p>Example: <code><F1></code> will open the Help file in most programs. <code><F2></code> in Windows Explorer will rename the currently selected file or folder. <code><F3></code> on Windows Explorer will open the Find Files dialog.</p>
<FIND ACTIVEWORDS[:AnyString]>

Opens the Find ActiveWords dialog to search for a desired string in the current WordBase. Optionally the command can include the search string as a parameter.

Example:

`<Find ActiveWords:<InputBox:"Search for ActiveWords containing:":Find ActiveWords>>` will popup an Input Box asking for the string to search and use it as a parameter to the Find ActiveWords command.

<GT>

<GREATER THAN>

Types the "greater than" character, ">", in a text substitution.

Example:

`<LT>ENTER<GT>` will type `<ENTER>`, rather than simulating the Enter key.

<HIDE DESKTOP ICONS>

Hide or show the icons of the desktop. There is no equivalent function available in windows.

<HIDE SHOW AWMONITOR>

Hide/show the Monitor bar. You will still be able to access the ActiveWords menu from the icon in the System Tray.

<HOME>

Simulates the Home key (goes to beginning of line or top of a list).

Example:

`<HOME>` in a word processor will put the cursor at the beginning of the current line.
`<CTRL><HOME></CTRL>` will go to the beginning of a document.

<"http://www.place.com[/somepath/somefile.html]">

This command will invoke your default Web Browser and navigate to the address specified. This is not an AWSL command but an HTML command supported by Windows, so the whole command including parameters must be enclosed in quotes as shown.

NOTE: This command will only work with Web Browser programs that support and are registered to handle the "http" protocol.

Example:

`<"http://www.ActiveWords.com">` will launch your default Web Browser and Navigate to the ActiveWords Web Site.

`<"http://www.activewords.com/purchasing.html">` will Navigate to the purchasing page at the ActiveWords Web Site.

<INPUTBOX[:Question[:Title]]>

Opens an Input Box that asks for information which will take the place of the InputBox command in the script. The action will then be executed as usual. The optional Question parameter will appear on the body and the Title parameter will appear on the header of the Input Box Dialog. The command can be used alone, with a Question or with a Question and a Title. [Pre-processed](#)

Example:

The following Script is a Navigation: `"<http://www.<INPUTBOX:Open a "COM" Internet Site. (Enter xxxx to go to www.xxxx.com):Go to Internet Site>.com/>"` which will pop up an Input Box with the phrase 'Open a "COM" Internet Site. (Enter xxxx to go to www.xxxx.com)' and with Go to Internet Site on the Title bar.

To include any syntax characters like : or > in the Question or Title, you can type the string in quotes, like `<INPUTBOX:"Go to folder:":<Folder Navigation>">` which will type "Go to folder:" (including the colon and without quotes) in the body and "<Folder

Navigation>" in the title bar.

If you need to begin the Body or the Title of an Input Box with quotes, but need to close the quotes before the end of the full body, precede the first quote with a space, like the title of the following example: `<INPUTBOX:"Type in your ToDo task:" "Quick" ToDo>`. This example will open an Input Box with the phrase "Type in your ToDo task:" (without quotes and including the ending colon), and the title ' "Quick" ToDo' with the word Quick in quotes.

<INSERT>

Simulates the Insert key.

<KEEP DIRECTORY OPEN> <KEEP GROUPBOX OPEN>

If the Action executed from a Directory contains this command, the Directory will NOT be automatically closed. [Pre-processed](#)

<LAST ACTIVEWORD>

Inserts the last ActiveWord triggered into the script. [Pre-processed](#)

<LAST DIRECTORY> <LAST GROUP>

Opens the last Directory displayed.

<LAST LINE[:##]>

Inserts the last line from the list of Last Typed Lines (LTL) A line is stored as an LTL each time the Enter key is pressed. The optional parameter in brackets specifies the ##th line from the LTL, where ## can be from 0 to 15, or the "Last Word" command (without the quotes), in which case the command tries to use the last word as a parameter and interpret it as a number. [Pre-processed](#)

Example:

`<LAST LINE>` will insert the line previous to the current line. Same as `<LAST LINE:0>`

`<LAST LINE:7>` will insert the seventh to last line typed by the user, starting from the last one up.

`<LAST LINE>Last Word>` will try to use the last word typed by the user before this ActiveWord as an index. If the Last Word is not a number, index 0 is used.

<LAST SPACED WORD[:##]>

Brings the Last Word typed by the user, considering only the SPACE and TAB as delimiters between words. For example 123-4567 is considered a Spaced Word even if the dash(-) is a delimiter, while for the <Last Word> command 4567 is the last word.

[Pre-processed](#)

Example:

After the following sentence: 12+5(19/5) Phone: 555-1234 2

Triggering `<LAST SPACED WORD>` will bring: 2

Triggering `<LAST SPACED WORD>Last Word>` which uses the number 2 as an index value will bring: Phone:

Triggering `<LAST SPACED WORD:3>` which uses the number 2 as an index value will bring: 12+5(19/5)

<LAST WORD[:##]>

Inserts the last word from the list of Last Typed Words (LTW). The optional parameter in brackets specifies the ##th word from the LTW, where ## can be from 0 to 15, or the <Last Word> command it self, in which case the command tries to use the last word as a parameter and interpret it as a number. [Pre-processed](#)

Example:

After the following sentence: My Phone number: 555-1234

Triggering `<LAST WORD>` will bring: 1234
 Triggering `<LAST WORD>` (the action triggered to run the command for `<LAST WORD:0>`) the word sentence for `<LAST WORD>Last Word>` as it took 1234 as index, but the maximum number allowed, 15, is used instead.

<LEFT[:##]>

Left arrow key. The optional parameter in brackets specifies the ## of times to press the Left arrow key.

Example:

`<LEFT:12>` will simulate pressing the Left arrow key twelve times.

<LOCATION:drawer:folder>

Opens a Directory with all ActiveWords contained in the specified Location. NOTE: This command is only intended for use in conjunction with the AW Development Center.

Example:

`<LOCATION:Drawer:Folder>` Shows all ActiveWords in the Drawer:Folder location.

`<LOCATION:Drawer:*>` Shows all ActiveWords from all folders of the Drawer location. (* serves as a wild card)

`<LOCATION:*:Folder>` Shows all ActiveWords contained in Folder, regardless of the Drawer.

<LT>

<LOWER THAN>

Types the "lower than" character, "<". Can be used to type a "lower than" character in a command so it is not interpreted as the beginning of a command.

Example:

`<LT>ENTER<GT>` will type `<ENTER>`, rather than simulating the Enter key.

<"Mailto: somebody@somewhere.com">

This command will invoke your default Email editor with the email address specified preset in the To: field. This is not an AWSL command but an HTML command supported by Windows, so the whole command including parameters must be enclosed in quotes as shown.

NOTE: This command will only work with Email programs that support and are registered to handle the HTML "Mailto" command.

Example:

`<"Mailto:sesupport@activewords.com">` will launch your default new Email composition program pre-addressed to sesupport@activewords.com.

<MAXIMIZE WINDOW>

Maximizes the current window. If this window was already maximized, this command has no effect. See also [<TOGGLE WINDOW SIZE>](#).

<MESSAGE BOX[:message[:title[:type]]]>

Displays a Message Box with an Ok button, pausing the current script until the user presses Ok. If used without parameters, this command will only play the default information (Asterisk) sound configured in Windows control Panel. The optional parameters are the message to be displayed in the body, the title of the Message Box and the type of icon to use on the box, which can be one of the following: INFORMATION, ERROR, EXCLAMATION, QUESTION. If the message and title are left blank, the type will specify the sound to play, no Message Box will be shown.

Example:

`<MESSAGE BOX:Pausing Script, press Ok to continue.:Script Pause...:Exclamation>`

Shows a Message Box with the text "Pausing Script, press Ok to continue." in the body, and "Script Pause..." as the title and the Exclamation mark icon.

To include a colon in the message or title, enclose it in quotes, like this:

`<MESSAGE BOX:"Press Ok when ready:":"Script Pause:">` This will show a message box with the "Press Ok when ready:" (without quotes) in the body and "Script Pause:" (without quotes) title and the Information icon (the default when omitted).

`<MESSAGE BOX:::Question>` will play the Question sound, without displaying a Message Box.

<MFC ABOUT> ([about MFC Commands](#))

Open the About box of current application.

<MFC ARRANGE ICONS>

Arrange the Icons of the current application.

<MFC CASCADE WINDOW>

Cascade all open Windows of current application.

<MFC FIND>

Open the find dialog box in current application.

<MFC NEW WINDOW>

Open a new window in current application.

<MFC PRINT>

Print the current document or file. Works only if the current application supports the Windows Print Command.

<MFC SAVE AS FILE>

Save the current file with a different name.

<MFC SAVE FILE>

Save the current file.

<MFC SPLIT WINDOW>

Split current Window.

<MFC TILE HORIZONTALLY>

Tile all open Windows of current application horizontally.

<MFC TILE VERTICALLY>

Tile all open Windows of current application vertically.

<MINIMIZE ALL>

Minimize all windows. Same as pressing WinStart+M

<MINIMIZE WINDOW>

Minimizes the current window. See also [<TOGGLE WINDOW SIZE>](#).

<MORE INFO>

Retrieves information related with the last ActiveWord triggered, from the Description field. [Pre-processed](#)

Example:

`<MORE INFO>` will type the contents of the Description field of the last item triggered.

<NAP AWMONITOR>

Turns off ActiveWords monitoring for the next typed word.

<NEXT WINDOW>

Navigate to the next window. Same as Alt+Esc.

<NOT:App1[,App2,...,AppN]>

Specifies that the Action should NOT be executed if the focus is on App1 [or App2 ... or AppN]. App1, App2...,AppN are the file names of the applications. To avoid a conflict among two programs with the same file name, a path name can be used. You can also use a part of the path name to prevent the execution of the ActiveWord on any programs that share this sub string in their path name. [Pre-processed](#)

Example:

<NOT:Winword.exe,Excel.exe> won't execute the Script if the current program is WinWord.exe or Excel.exe.

<NOT:Microsoft Office> will prevent the Script from being executed on any of the Microsoft Office suite of applications, because they all share the substring "Microsoft Office" in their path name, commonly "C:\Program Files\Microsoft Office\Office\ ..."

<NULL>

NULL action, do nothing.

<ONLY:App1[,App2,...,AppN]>

Specifies that the Action should be executed ONLY if the focus is on App1 [or App2 ... or AppN]. App1, App2...,AppN are the file names of the applications. To avoid a conflict among two programs with the same file name, a path name can be used. You can also use a part of the path name to allow the execution of the ActiveWord on all the programs that share this sub string in their path name. [Pre-processed](#)

Example:

<ONLY:Winword.exe,Excel.exe> will execute the Script only if the current program is WinWord.exe or Excel.exe.

<ONLY:Microsoft Office> will let the Script to be executed on all the Microsoft Office suite of applications, because they all share the sub string "Microsoft Office" in their path name, commonly "C:\Program Files\Microsoft Office\Office\ ..."

<PAGE DOWN>

Simulates the Page Down key.

<PAGE UP>

Simulates the Page Up key.

<PASSWORD MODE AWMONITOR>

Sets/removes ActiveWords from password mode. When in Password Mode, the AWMonitor will display asterisks (*) as the user types, but ActiveWords will still work.

<"PATH">

The Path of any executable or file can be included as part of a script by enclosing the full path name in quotes and chevrons (< >). For most executables and files, the execution of the script will pause until the application is up and running, then the script following the Path, if any, will be executed. Parameters passed to the application can also be used. As long as the path starts with quotes, ActiveWords will consider everything until the next closing chevron (>) as part of the execution.

Examples:

The following script will start Notepad and insert the current date:

<"C:\Windows\notepad.exe"><TAB:8> <F5><ENTER>

There are certain programs that require their parameters to be used without quotes. For example, the following script will open Netscape Messenger and create a new mail message. This script also uses the preprocessed command <ProgFilesDir> which will be replaced by the Windows Program files folder (usually C:\Program Files):

```
<"<ProgFilesDir>\Netscape\Communicator\ Program\Netscape.exe" -
mail> <Ctrl>m</Ctrl>
```

Note: In order for this syntax to function properly, the ActiveWord must be saved as a Command. If you used the browse function on AWDevelopmentCenter to add the path of the file, and then edited the script to enclose the path in chevrons (< >), you must use the Expert function from the toolbar to manually set the Action type to Command.

<PAUSE>

Simulates the Pause key.

<PREVIOUS WINDOW>

Navigate to the previous window. Same as Alt+Tab

<PRODUCTIVITY>

Opens the Productivity Center dialog.

<PROGFILES DIR>

<PROGRAMFILES DIR>

<PROGRAM FILES DIR>

The complete "Program Files" path is returned (i.e. "C:\Program Files") regarding the language of the Operating System. [Pre-processed](#)

Example:

The following Navigation item:

```
"<PROGRAM FILES DIR>\Accessories\WordPad.exe" will open Microsoft WordPad.
```

<REGISTRY VALUE:Root±Key±Name>

The value of the specified Registry entry is returned, if the entry is found. [Pre-processed](#)
If the Registry entry does not exist, the script is not executed to prevent undesirable behavior.

Root: Can be any of the following root values of the Registry

```
HKEY_CLASSES_ROOT
HKEY_CURRENT_CONFIG
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_PERFORMANCE_DATA
HKEY_DYN_DATA
```

Key: Specifies the path, or key of the registry value that contains the information to retrieve, without the root value (located on the left side of the Registry Editor).

Name: Specifies the name of the value that contains the information to retrieve (located on the right side of the Registry Editor). If no name is specified, the "Default" value is used.

Example:

The following Command item.

```
<"<REGISTRY VALUE: HKEY_LOCAL_MACHINE
±Software\Microsoft\Windows\CurrentVersion\App
Paths\WINWORD.EXE±Path>\Winword.exe">
```

will open Microsoft Word, regardless of its installation directory. If Microsoft Word is not installed (its registry key does not exist) the command will not execute.

<RESTORE WINDOW>

Restore window from maximized to restored view. If the window was already restored, this command has no effect. See also [<TOGGLE WINDOW SIZE>](#).

<RIGHT[: ##]>

Simulates the Right arrow key. The optional parameter in brackets specifies the ## of times to press the Right arrow key.

Example:

`<RIGHT:7>` will simulate pressing the Right arrow key 7 times.

<SAVE TEMPORAL ACTIVEWORDS>

Saves temporal ActiveWords through the corresponding Editor.

<SCREEN SAVER>

Starts the default Windows Screen Saver.

<SHIFT> </SHIFT>

`<SHIFT>` simulates the Shift key down, `</SHIFT>` simulates the Shift key up. A `<SHIFT>` must always be closed by a `</SHIFT>`.

Example:

`<SHIFT><INSERT></SHIFT>` will paste the contents of the clipboard.

<SLEEP WAKE AWMONITOR>

Turns ActiveWords monitoring on/off. While ActiveWords monitoring is off the only commands that will work are `<SLEEP WAKE AWMONITOR>` and `<CLOSE AWMONITOR>`

<SPACE[: ##]>

Simulates the Spacebar key. The optional parameter in brackets specifies the ## of times to press the Spacebar key.

Example:

`<SPACE:10>` will simulate pressing the Spacebar ten times.

<SPECIAL FOLDER:Folder name>

<FOLDER:Folder name>

The complete path of the specified Special Folder is returned (i.e. "C:\Windows\Start Menu\Programs") regarding the language of the Operating System. [Pre-processed](#)
Where Folder name can be any of the following:

APPDATA
BITBUCKET
COMMON_DESKTOPDIRECTORY
COMMON_PROGRAMS
COMMON_STARTMENU
COMMON_STARTUP
DESKTOPDIRECTORY
FAVORITES
NETHOOD
PERSONAL
PRINTHOOD
PROGRAMS
RECENT
SENDTO
STARTMENU
STARTUP
TEMPLATES

Example:

The following Navigation item

`"<FOLDER:PROGRAMS>\Accessories\Notepad.Ink"` will open the shortcut that points to Notepad from the StartMenu\Programs\Accessories menu.

<SUBMIT:word>

Submits the specified word just as if the user typed it. ActiveWords will present the Word Detected banner and message and will wait for a confirmation (typically a space) to execute the action associated with the word. This command must be at the end of any script in order to work correctly.

Example:

`<SUBMIT:notepad>` will ask for a confirmation to run notepad.

<TAB[:##]>

Simulates the Tab key. The optional parameter in brackets specifies the ## of times to press the Tab key.

Example:

`<TAB:3>` will simulate pressing the Tab key 3 times.

`<ALT><TAB></ALT>` will navigate to the previous open Window.

<TOGGLE WINDOW SIZE>

If the foremost window is maximized, `<TOGGLE WINDOW SIZE>` will restore it, if it is restored, it will maximize it.

<UP[:##]>

Simulates the Up arrow key. The optional parameter in brackets specifies the ## of times to press the Up arrow key.

Example:

`<UP:2>` will simulate pressing the Up arrow key twice.

<USER/PROFILE>

Opens the Users/Profiles dialog to change or add a user or profile.

<WAIT FOR WINDOW:TITLE=Window Title[:TIMEOUT=TimeOut]>

<WAIT FOR WINDOW:CLASS=Class Name[:TIMEOUT=TimeOut]>

Pauses the execution of the current script and looks among all open Windows for one which contains the Window Title specified or the Class Name specified. When such Window appears or if it already exists, ActiveWords will bring it to current focus and then continue the rest of the script. A TimeOut period of up to 60 seconds can be specified until the command aborts if no such Window is found. The default is 10 seconds if no TimeOut is specified.

Example:

`<WinStart>f</WinStart><WAIT FOR WINDOW:TITLE=Find>Notepad <Enter>` will invoke the Windows Find Files dialog and wait (up to 10 seconds) until the dialog appears (which title contains the Find string). Then it will type Notepad and press the Enter key to begin searching.

<WINDIR>

<WINDOWS DIR>

<WINDOWS DIR>

The complete Windows path is returned (i.e. "C:\Windows" on Windows 95 or "D:\WinNT" on Windows NT) regarding the language of the Operating System. See [<PROGRAM FILES DIR>](#) for an example. [Pre-processed](#)

<WINSYSDIR>

<WINDOWSSYSTEMDIR>

<WINDOWS SYSTEM DIR>

The complete Windows System path is returned (i.e. "C:\Windows\System" on Windows 95 or "D:\WinNT\System32" on Windows NT) regarding the language of the Operating System. See [<PROGRAM FILES DIR>](#) for an example. [Pre-processed](#)

<WINMENU>

Simulates a right mouse click in Windows to open a Context Menu.

Example:

`<WINMENU>r` on the Desktop will open the Display Properties Dialog.

<WINSTART> </WINSTART>

Activates the Windows "START" menu. A `<WINSTART>` must always be closed by a `</WINSTART>`.

Example:

`<WINSTART></WINSTART>` will activate the "Start" button on the Task Bar.

`<WINSTART>f</WINSTART>` will open the Find Files dialog.

`<WINSTART>e</WINSTART>` will open Windows Explorer.

`<WINSTART>r</WINSTART>` will open the Run dialog.

<WORD:word[:Stop | :Continue]>

A *word* is the ActiveWord assigned to any item where the Confirmation Flag is checked (any ActiveWord already existing in the user's WordBase with the Confirmation Flag checked). You can use chains of *words* in a Script and each nested *word's* script can also include embedded *words*. You can use as many chained *words* as you want in a script. You can also use up to ten levels of recursion, where *word1* calls *word2*, *word2* calls *word3* ... *word9* calls *word10*.

Example:

Consider a group of 3 users in a computer network. The administrator has an ActiveWord to send a message to each one of the users, Send1 for user 1, Send2 for user 2 and so on. The Confirmation Flag is checked for each of these ActiveWord items. The following script would send the message to all 3 users:

ActiveWord: `SendAll`

Action: `<WORD:send1><WORD:send2><WORD:send3>`

You can combine the use of embedded ActiveWords with the conditional commands `<Only>`, `<Not>` and `<InputBox>` by using the `:Stop` and `:Continue` parameters in the following way:

If the `:Stop` parameter is specified, the rest of the Script that follows the embedded ActiveWord won't be executed upon successful completion of the embedded ActiveWord's script. That is, if the embedded ActiveWord's script has the command `<Only:App1>`, the rest of the calling Script will stop if the current application is App1; if the embedded ActiveWord's script has the command `<Not:App1>`, the rest of the calling Script will stop if the current application is not App1; if the embedded ActiveWord's script has the command `<InputBox>`, the rest of the calling Script will stop if the user presses Ok on the InputBox.

If the `:Continue` parameter is specified, the rest of the Script that follows the embedded ActiveWord will be executed upon successful completion of the embedded ActiveWord's script. That is, if the embedded ActiveWord's script has the command `<Only:App1>`, the rest of the calling Script will continue if the current application is App1; if the embedded ActiveWord's script has the command `<Not:App1>`, the rest of the calling Script will continue if the current application is not App1; if the embedded ActiveWord's script has the command `<InputBox>`, the rest of the calling Script will continue if the user presses Ok on the InputBox.

Example:

Consider the following ActiveWords:

ActiveWord 1: Word: `NPadFind`

Action: `<ONLY:Notepad.exe><ALT>s</ALT>f`

ActiveWord 2: Word: `ExpFind`

```
Action: <ONLY:Explorer.exe><WINSTART>f</WINSTART>
ActiveWord 3: Word: Find
Action: <WORD:NPadFind:Stop><WORD:ExpFind:Stop><CTRL>f</CTRL>
```

Note that ActiveWord number 3 uses the ActiveWords 1 and 2 embedded with the :Stop parameter. Also, both ActiveWords 1 and 2 use the command <Only:...> in their respective scripts. This gives us the following possible scenarios:

- If you run ActiveWord 3 on Notepad.exe, the embedded ActiveWord NPadFind will be successfully executed because the <ONLY:Notepad.exe> command will return TRUE. The rest of the Script in ActiveWord 3 (ExpFind and Ctrl+F) will not be executed because NPadFind has the :Stop parameter.
- If you run ActiveWord 3 on any application but Notepad.exe or Explorer.exe, ActiveWord 1 and ActiveWord 2 won't be executed (because their corresponding <Only:...> clause will return FALSE) and the fragment <CTRL>f</CTRL> from ActiveWord 3 will be executed.

This example demonstrates a Generic Find ActiveWord.

<WORK PAD[:Title]>

Displays a dialog box into which information may be entered. An optional title can be specified.

Example:

```
<WORK PAD:Example>Hi there!!!
```

Will open the Work Pad dialog window with "Example" in its title bar and type "Hi there!!!" within the window.

[Back to top](#)

Notes:

- 1 Parameters between brackets ([]) are optional. The brackets are not part of the command syntax and should be removed for the parameter to be properly recognized.
- 1 The maximum number of key repetitions ([:##]) for the commands that support it is 200.
- 1 **Pre-processed:** The command is processed before beginning the execution of the script so it can be used on any type of ActiveWord (Substitutions, Commands or Navigations) and can be placed anywhere in the Action script. On Substitutions this command will only function if the Action is less than 255 characters (including the command itself).
- 1 Please note that commands having a closing counterpart, like <ALT></ALT>, require the closing part to function. The closing part is always represented by a preceding slash / (using a back-slash \ will not work). The opening part of these commands simulates pressing a key. The closing part simulates releasing a key.
- 1 **MFC commands** will work only on MFC (Microsoft Foundation Classes) compliant applications and only if the application or the application's current view supports the command.