WshShell.SendKeys Method

Description

The **SendKeys** method sends one or more keystrokes to the active window (as if typed on the keyboard).

Syntax

```
object.SendKeys(string)
```

Arguments

Parameter	Description
string	String value indicating the keystroke(s) you want to send.

Notes

- Use the **SendKeys** method to send keystrokes to applications that have no automation interface.
- Most keyboard characters are represented by a single keystroke.
- Some keyboard characters are made up of combinations of keystrokes (CTRL+SHIFT+HOME, for example). To send a single keyboard character, send the character itself as the string argument.
- To send a space, send the string " ".
- You can use **SendKeys** to send more than one keystroke at a time. To do this, create a compound string argument that represents a sequence of keystrokes by appending each keystroke in the sequence to the one before it. For example, to send the keystrokes a, b, and c, you would send the string argument "abc".
- The **SendKeys** method uses some characters as modifiers of characters (instead of using their face-values).
  - plus sign "+"
  - caret "^"
  - percent "%"
  - tilde "~"
- Send these characters by enclosing them within braces "{}". For example, to send the plus sign, send the string argument "{+}". Brackets "[ ]" have no special meaning when used with SendKeys, but you must enclose them within braces to accommodate applications that do give them a special meaning (for dynamic data exchange (DDE) for example).
- To send bracket characters, send the string argument "{[}" for the left bracket and "{]}" for the right one.
- To send brace characters, send the string argument "{{}" for the left brace and "{}}" for the right one.
- Some keystrokes do not generate characters (such as ENTER and TAB). Some keystrokes represent actions (such as BACKSPACE and BREAK). To send these kinds of keystrokes, send the arguments shown in the following table:

Key	Argument	Key	Argument

BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}	F1	{F1}
BREAK	{BREAK}	F2	{F2}
CAPS LOCK	{CAPSLOCK}	F3	{F3}
DEL or DELETE	{DELETE} or {DEL}	F4	{F4}
DOWN ARROW	{DOWN}	F5	{F5}
END	{END}	F6	{F6}
ENTER	{ENTER} or ~	F7	{F7}
ESC	{ESC}	F8	{F8}
HELP	{HELP}	F9	{F9}
HOME	{HOME}	F10	{F10}
INS or INSERT	{INSERT} or {INS}	F11	{F11}
LEFT ARROW	{LEFT}	F12	{F12}
NUM LOCK	{NUMLOCK}	F13	{F13}
PAGE DOWN	{PGDN}	F14	{F14}
PAGE UP	{PGUP}	F15	{F15}
PRINT SCREEN	{PRTSC}	F16	{F16}
RIGHT ARROW	{RIGHT}	TAB	{TAB}
SCROLL LOCK	{SCROLLLOCK}	UP ARROW	{UP}

To send keyboard characters that are comprised of a regular keystroke in combination with a SHIFT, CTRL, or ALT, create a compound string argument that represents the keystroke combination. You do this by preceding the regular keystroke with one or more of the following special characters:

Key	Special Character
SHIFT	+
CTRL	^
ALT	%

- When used this way, these special characters are not enclosed within a set of braces.
- To specify that a combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, create a compound string argument with the modified keystrokes enclosed in parentheses. For example, to send the keystroke combination that specifies that the SHIFT key is held down while:
  - **e** and **c** are pressed, send the string argument "**+(ec)**".
  - **e** is pressed, followed by a lone **c** (with no SHIFT), send the string argument "**+ec**".
- You can use the **SendKeys** method to send a pattern of keystrokes that consists of a single keystroke pressed several times in a row. To do this, create a compound string argument that specifies the keystroke you want to repeat, followed by the number of times you want it repeated. You do this using a compound string argument of the form {keystroke number}. For example, to send the letter "x" ten times, you would send the string

argument "{x 10}". Be sure to include a space between keystroke and

number.

- The only keystroke pattern you can send is the kind that is comprised of a single keystroke pressed several times. For example, you can send "x" ten times, but you cannot do the same for "Ctrl+x".
- You cannot send the PRINT SCREEN key {PRTSC} to an application.

Example

I Find that **SendKeys** method is very useful in **QuickTest**, I use it a lot, especially on "no Web applications".

I have experience that sometimes when selecting a menu-item, is not always activated.

i.e. `Window("Notepad").WinMenu("Menu").Select "File;New Ctrl+N"`

In other applications the menu is not recognized by **QuickTest**, in WEB usually the menu is a collection of Web-Elements.

Every Windows application should support Keyboard usage, so, why don't we use it?

In the following sample I will show you how to handle all the menus in notepad, using the **SendKeys** method, and implement all the menu in a one single line.

External vbs/qfl file.

```
Option Explicit
RegisterUserFunc "Window", "SelectMenuItem", "MySelectMenuItem"
Public Sub MySelectMenuItem( ByVal obj, ByVal sMenuItem)
    Dim oShell
    Dim sSendKey
    Select Case LCase(sMenuItem)
    Case "about notepad"      : sSendKey = "%ha"
    Case "help topics"        : sSendKey = "%hs"
    Case "status bar"         : sSendKey = "%vs"
    Case "font"               : sSendKey = "%of"
    Case "word wrap"          : sSendKey = "%ow"
    Case "time date"          : sSendKey = "{F5}"
    Case "select all"         : sSendKey = "^a"
    Case "go to"              : sSendKey = "^g"
    Case "replace"            : sSendKey = "^h"
    Case "find next"          : sSendKey = "{F3}"
    Case "find"               : sSendKey = "^f"
    Case "delete"             : sSendKey = "{DEL}"
    Case "paste"              : sSendKey = "^v"
    Case "copy"               : sSendKey = "^c"
    Case "cut"                : sSendKey = "^x"
    Case "undo"               : sSendKey = "^u"
    Case "exit"               : sSendKey = "%fx"
    Case "print"              : sSendKey = "^p"
```

```
    Case "page setup"          : sSendKey = "%fu"
    Case "save as"             : sSendKey = "%fa"
    Case "save"                : sSendKey = "^s"
    Case "open"                : sSendKey = "^o"
    Case "new"                 : sSendKey = "^n"
    Case Else
         Reporter.ReportEvent micFail, "SelectMenuItem", _
```



```
                              "Menu item '" & sMenuItem & "' does not exist."
         Err.Raise VbObjectError + 1002, "SelectMenuItem",  _
                              "Menu item '" & sMenuItem & "' does not exist."
         Exit Sub
    End Select
    Set oShell = CreateObject("WScript.Shell")
    obj.Activate : Wait 0, 200
    oShell.SendKeys sSendKey
    Set oShell = Nothing
End Sub
```

- After the function is ready, move the function to an external **vbs** file.
- Associate the **vbs** file to the test setting resources in **QuickTest**; Test->Settings->Resources->Add a new file to the list and select the **vbs** file you have just created.
- Of course we can call this function from our script using the follow syntax: