GetState

```
a:= Act{WindHand, "GetState"};
```

Returns the check state of a radio button or check box. The return is 0 for unchecked, 1 for checked or 2 for indeterminate. The Indeterminate state only applies to a few 3 state buttons.

WindHand	Either the handle of the button window or a description of the Button window. See the FindWindow function for acceptable descriptions.

SetState

```
Act{WindHand,"SetState", NewState};
```

Sets the state of a radio button or check box.

WindHand	Either the handle of the button window or a description of the Button window. See the FindWindow function for acceptable descriptions.
NewState	The new state of the button window. 0 = unchecked, 1=checked, 2= unknown. The unknown (or indeterminate) state only applies to 3 state check boxes.

Press

```
Act{WindHand, "Press"};
```

Presses a push button (for example pressing the 'OK' button is the same as clicking on it).

WindHand	Either the handle of the button window or a description of the Button window. See the FindWindow function for acceptable descriptions.

GetSelected

```
a:=Act{WindHand,"GetSelected"};
```

Returns the index of the selected item from within the listbox section of a combo box. The index starts at 0 (the top item in the list). If there is no selected item, this function returns -1.

WindHand	Either the handle of the combo box window or a description of the combo box window. See the FindWindow function for acceptable descriptions.

Note

You can find out how many items are in a combo box by using GetComboText to obtain an array of all the text in the combo box and using the size function to find how many items there are.

SetSelected

```
Act{WindHand,"SetSelected",NewSel};
```

Sets the selected item within the list box section of a combo box.

WindHand	Either the handle of the combo box window or a description of the combo box window. See the FindWindow function for acceptable descriptions.
NewSel	The new item to select (0=the first item). If the combo box contains n items them the last item is number n-1.

See GetSelected for how to find the number of items in a combo box.

GetText

```
a:=Act{WindHand,"GetText"}
```

Returns the text from within the edit section of a combo box.

WindHand	Either the handle of the combo box window or a description of the combo box window. See the FindWindow function for acceptable descriptions.

SetText

```
Act{WindHand,"SetText", NewText};
```

Sets the text contained in the edit section of a combo box. Note that the combo box muts be editable for this function to work. If the combo box is a simple drop down list this function does nothing.

WindHand	Either the handle of the combo box window or a description of the combo box window. See the FindWindow function for acceptable descriptions.
NewText	A string containing the new text.

Example:

```
Act{WindHand,"SetText","Hello"};
```

Places the word "Hello" into the edit window of a combo box.

GetComboText

```
a:=Act{WindHand, "GetComboText"};
```

Returns a one dimensional array containing all the text from within the list section of a Combo Box.

WindHand	Either the handle of the combo box window or a description of the combo box window. See the FindWindow function for acceptable descriptions.

Example:

```
a:=Act{WindHand, "GetComboText"};
for (i:=0; i<size(a); i+=1;) {
   MessageBox(a[i]);
};
```

This fragment of code gets the text from a combo box (whose handle is WindHand) and then displays a message box for each line of text it found.

SetText (Edit Window)

```
Act{WindHand, SetText, New_String };
```

Window Act to set the text within an Edit window, replacing any previous text.

WindHand	Either the handle of the edit window or a description of the Edit window. See the FindWindow function for acceptable descriptions.
New_String	A string object containing the new text.

Notes

To enter multiple lines into a multi-line edit window, use carriage return (character 13) and line feed characters (character 10).

For example

```
 Act{EditWind, SetText,This is text+13+10+This is line 2};
```

This example enters two lines of text into an edit control.

See Also

GetText - Returns the text in an Edit Control

Window Acts - A General overview of Window Acts.

GetText

```
a := Act{WindHand,"GetText"};
```

Returns the text contained within a single or multi-line edit control as a string object.

WindHand	Handle of the Edit window or a description of the Edit window using any of the methods listed under the FindWindow function.

Example:

```
a := Act{{"Notepad",">edit"}, "GetText"};
```

Returns the text from within the Edit Window inside the Notepad. Where

See Also

SetText - Sets the text within an Edit Control

Window Acts - A General overview of Window Acts.

GetLineCount

```
a := Act{WindHand, "GetLineCount"};
```

Returns the number of lines of text contained within an Edit Window. An empty edit window still returns 1 line.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.

Example:

a := Act{`{"Notepad",">edit"},"GetLineCount"};`

GetSel

```
a := Act{WindHand, "GetSel"};
```

Returns the selected area of text as an array containing two elements. The array looks like this: {Start_Sel, End_Sel} where Start_Sel and End_Sel are the offsets (measured in characters) from the start of the text.

If there is no selected text the start and end positions are the same.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.

Example:

a := Act{`{"Notepad",">edit"},"GetSel"};`

returns the array {3,3} indicating that there is no selected text and that the cursor is at character 3 in the text.

Paste

```
Act{WindHand, "Paste"};
```

Pastes text from the clipboard into the specified Edit Window. This is exactly the same as selecting Paste from a menu.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.

Example

```
SetClipboardText("Some Sample Text");
a := Act{WindHand, "Paste"};
```

Pastes "Some Sample Text" into the notepad.

See Also

Window Acts - A General overview of Window Acts.

Copy

```
Act{WindHand, "Copy"};
```

Copies any selected text from the Edit window to the clipboard.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.

Example

```
Act{WindHand, "Copy"};
a:=GetClipboardText;
```

See Also

Window Acts - A General overview of Window Acts.

Cut

```
Act{WindHand,"Cut"};
```

Cuts any selected text from the specified Edit window to the clipboard.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.

Example

```
Act{WindHand, "Cut"};
a:=GetClipboardText;
```

See Also

Window Acts - A General overview of Window Acts.

Clear

```
Act{WindHand,"Clear"};
```

Deletes any selected text from the specified Edit window and discards it.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.

Example

```
Act{WindHand, "Clear"};
```

See Also

Window Acts - A General overview of Window Acts.

Undo

```
Act{WindHand, "Undo"};
```

Undoes the last action performed on the Edit window.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.

Example

```
Act{WindHand, "Undo"};
```

See Also

Window Acts - A General overview of Window Acts.

SetReadOnly

```
Act{WindHand, "SetReadOnly", Flag}
```

Sets or clears the read only flag. If the flag is TRUE, then the text in the edit window can only be viewed, it cannot be edited.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.
Flag	A Boolean TRUE or FALSE, TRUE means read only. FALSE means read-write.

Example

```
Act{WindHand, "Undo"};
```

See Also

Window Acts - A General overview of Window Acts.

SetSel

```
Act{WindHand, "SetSel", {StartSel, EndSel}}
```

Sets the selected region of an Edit window or places the cursor.

WindHand	Either the handle of the Edit Window, or any method of describing the Edit Window as listed under FindWindow.
StartSel	The start of the selected text as an offset in characters from the start of the text.
EndSel	The end of the selected text as an offset in characters from the start of the text.

Note that StartSel <= EndSel. There is however a special case {0,-1} which always selects all the text in an Edit window.

Examples

```
Act{WindHand, "SetSel", {0, 4}}
```
Sets the selection to run from character 0 to character 4.
```
Act{WindHand, "SetSel", {4, 4}}
```
Sets the cursor position to character 4
```
Act{WindHand,"SetSel",{0,-1}}
```
Selects all the text in an Edit window.

GetSelected

```
a:=Act{WindHand,"GetSelected"};
```

Returns the index of the currently selected item from a single selection list box. The indices start at 0 for the top item in the list.

WindHand	Either the handle of the list box window or a description of the list box window. See the FindWindow function for acceptable descriptions.

You can find out how many items are in a list box by using GetListBoxText to obtain an array of all the text in the list box and using the size function to find how many items there are.

This command work differently for a multi-selection list box. See GetSelected.

SetSelected

```
Act{WindHand, "SetSelected", Index};
```

Selects the item in the list box whose position is given by Index. (Index starts at 0 for the top item in the list box). Any previously selected item is deselected.

```
Act{WindHand, "SetSelected", MatchString};
```

In this form it searches for a match (or partial match) to MatchString and selects that.

WindHand	Either the handle of the list box window or a description of the list box window. See the FindWindow function for acceptable descriptions.

Examples:

```
Act{WindHand,"SetSelected",2};
```

Selects item number 2 (the third item in the list since 0 is the first item).

```
Act{WindHand,"SetSelected", "Currency Format"};
```

Searches for an item "Currency Format" in the list box and selects it.

This command work differently for a multi-selection list box. See SetSelected.

GetListBoxText

```
a:=Act{WindHand, "GetListBoxText"};
```

Returns an array containing the text from within a list box. Note: Special owner drawn list boxes do not contain text.

WindHand	Either the handle of the list box window or a description of the list box window. See the FindWindow function for acceptable descriptions.

A typical return from this function looks like this:

{"Red","Green","Blue","Yellow","Orange","Purple"};

GetSelected

```
a:=Act{WindHand,"GetSelected"};
```

Returns an array of indices for each selected item in the list box. The indices start at 0 for the top item in the list.

WindHand	Either the handle of the list box window or a description of the list box window. See the FindWindow function for acceptable descriptions.

A typical return from this function looks like: {3@, 4@, 5@} Which indicates 3 selected items at position 3, 4 and 5 (@ in Flute indicates an integer).

You can find out how many items are in a list box by using GetListBoxText to obtain an array of all the text in the list box and using the size function to find how many items there are.

This command work differently for a single-selection list box. See GetSelected.

SetSelected

```
Act{WindHand, "SetSelected", Index};
```

Toggles the selection state of the item at position 'Index'. A selected item becomes deselected and visa-versa.

```
Act{WindHand, "SetSelected", {Index, Flag}};
```

In this syntax the item at position 'Index' is either selected or deselected. If Flag is TRUE the item is selected, FALSE will deselect the item.

WindHand	Either the handle of the list box window or a description of the list box window. See the FindWindow function for acceptable descriptions.
Index	The position of the item, 0 = top, 1,2.. Or a text string. If you provide a text string, SetSelected will search for a complete or partial match for the string in the list box.
Flag	Boolean TRUE or FALSE, (or numerical 0 is treated as FALSE and non-zero as TRUE)

This command work differently for a single-selection list box. See SetSelected.

Examples:

```
Act{WindHand, "SetSelected", 2}
```

Changes the selection state of item 2 in the listbox.

```
Act{WindHand,"SetSelected", {"Currency Format", TRUE}};
```

Selects the 'Currency Format' item in the list box.

Activate

```
Act{WindHand,"Activate",WindName};
```

Activates the child window called *WindName*. The name is not case sensitive, but must otherwise be identical to the title of the window. The active window receives input from the keyboard.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.
WindName	A string object that contains the name of the window to activate, for example "Sheet1.cln".

Cascade

```
Act{WindHand, "Cascade"};
```

Cascades the child windows of the MDI Client window. This is the same as selection 'Cascade' on the 'Windows' menu of the application.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.

GetActive

```
a:=Act{WindHand, "GetActive"};
```

Returns the handle of the active window out of the MDI child windows.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.

ArrangeIcons

```
Act{WindHand, "ArrangeIcons"};
```

Arranges the icons in the window. It does not affect windows that are not iconized.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.

Maximize

```
Act{WindHand, "Maximize", WindName};
```

Maximizes the named child window.

```
Act{WindHand, "Maximize"};
```

In this form (omitting the name of the window) the presently active window of the group is maximized.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.
WindName	Name of the child window to maximize as a string object. This name must have exactly the same punctuation and characters as the window's title, but the case is ignored. (e.g. "Untitled1:2").

Restore

```
Act{WindHand, "Restore", WindName};
```

Restores the named child window to its default size (a restored window is one that is neither maximized or minimized).

```
Act{WindHand, "Restore"};
```

In this form (omitting the name of the window) the presently active window of the group is restored.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.
WindName	Name of the child window to restore. This name must have exactly the same punctuation and characters as the window's title, but the case is ignored. (e.g. "Untitled1:2").

TileHorz

```
Act{WindHand, "TileHorz"};
```

Tiles the windows horizontally. When there are many windows to tile, the program tiles vertically and horizontally.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.

TileVERT

```
Act{WindHand, "TileVert"};
```

Tiles the windows vertically. When there are many windows to tile, the program tiles vertically and horizontally.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.

ChildWindows

```
a := Act{WindHand, "ChildWindows"};
```

Returns an array of the child windows present under the MDI Client window. These are usually worksheets or document windows.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.

The return appears as an array of integer objects, for example: {3231@, 23645@,19284@} indicates 3 child windows.

Minimize

```
Act{WindHand, "Minimize", WindName};
```

Minimizes (iconizes) the named child window.

```
Act{WindHand, "Minimize"};
```

In this form (omitting the name of the window) the presently active window of the group is minimized.

WindHand	Either the handle of the MDI Client window or a description of the client window. See the FindWindow function for acceptable descriptions.
WindName	Name of the child window to minimize as a string object. This name must have exactly the same punctuation and characters as the window's title, but the case is ignored. (e.g. "Untitled1:2").

LineUp

```
Act{WindHand, "LineUp"};
```

Scrolls one line up on a vertical scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar window. See the FindWindow function for acceptable descriptions.

LineDown

```
Act{WindHand, "LineDown"};
```

Scrolls one line down on a vertical scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

PageUp

```
Act{WindHand, "PageUp"};
```

Scrolls one page up on a vertical scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

PageDown

```
Act{WindHand, "PageDown"};
```

Scrolls one page down on a vertical scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

SetPos

```
Act{WindHand, "SetPos", NewPos};
```

Moves the *thumb* of the scroll bar to a new position. The thumb is the box that indicates the position of the scroll bar.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.
NewPos	The new position of the scroll bar as a number 0 to 100 where 0 is the top/left position and 100 is the bottom/right.

GetPos

```
a:=Act{WindHand, "GetPos"};
```

Returns the position of the scroll bar thumb as a number 0-100, where 0 is the top/left and 100 is the bottom/right of the scroll bar.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

LineLeft

```
Act{WindHand, "LineLeft"};
```

Scrolls one line left on a horizontal scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

LineRight

```
Act{WindHand, "LineRight"};
```

Scrolls one line right on a horizontal scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

PageLeft

```
Act{WindHand, "PageLeft"};
```

Scrolls one page left on a horizontal scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

PageRight

```
Act{WindHand, "PageRight"};
```

Scrolls one page right on a horizontal scroll bar control.

WindHand	Either the handle of the scroll bar window or a description of the scroll bar box window. See the FindWindow function for acceptable descriptions.

GetText

```
a := Act{WindHand, "GetText"};
```

Returns the text contained in a status bar as an array of strings.

WindHand	Either the handle of the status window or a *description* of the status window. See the FindWindow function for acceptable descriptions.

For example, a typical return from the Explorer status window is:

{"87 Object(s)", "1.82Mb"}

GetPos

```
a:=Act{WindHand, "GetPos"};
```

Returns the position of the 'thumb' on the trackbar. The position is returned as a double value in the range 0 to 100 (0 is left or top and 100 is right or bottom).

WindHand	Either the handle of the track bar window or a *description* of the track bar window. See the FindWindow function for acceptable descriptions.

SetPos

```
Act{WindHand, "SetPos", Pos};
```

Sets the position of the thumb in the trackbar.

WindHand	Either the handle of the track bar window or a *description* of the track bar window. See the FindWindow function for acceptable descriptions.
Pos	The new position as a numeric value in the range 0 (left or top) to 100 (right or bottom).

GetSelRange

```
Range := Act{WindHand, "GetSelRange"};
```

Returns the selection range for a trackbar that supports range selection. The range is returned as a two element array {StartRange, EndRange} where StartRange and EndRange are numbers from 0 (left or top) to 100 (right or bottom).

WindHand	Either the handle of the track bar window or a *description* of the track bar window. See the FindWindow function for acceptable descriptions.

SetSelRange

```
Act{WindHand, "SetSelRange", {StartRange, EndRange}};
```

Sets the selected range for TrackBars that support selections.

WindHand	Either the handle of the track bar window or a *description* of the track bar window. See the FindWindow function for acceptable descriptions.
StartRange	The start of the selected range as a number 0 to 100 (0 is left/top of trackbar, 100 is right/bottom)
EndRange	The end of the selected range as a number 0 to 100.

For Example:

```
Act{{"$media player",">msctls_trackbar32"},"SetSelRange",{33.3,66.6}};
```

Sets the selected range for the media player to the middle 1/3rd of its range (33.3 to 66.6).

Up

```
Act{WindHand, "Up"};
```

Equivalent to clicking on the up arrow of a spin dial control.

WindHand	Either the handle of the spin dial window or a description of the spin dial window. See the FindWindow function for acceptable descriptions.

Down

```
Act{WindHand, "Down"};
```

Equivalent to clicking on the down arrow of the spin dial.

WindHand	Either the handle of the spin dial window or a description of the spin dial window. See the FindWindow function for acceptable descriptions.