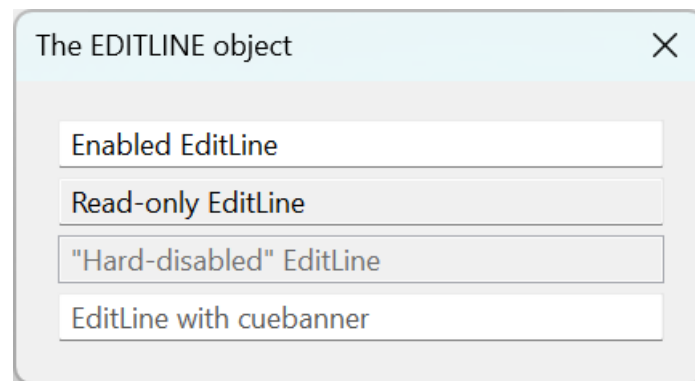


EDITLINE object

The EDITLINE is the basic single-line OpenInSight text-entry control and is used for ordinary text, constrained input, password entry and Windows autocomplete.



Developer Notes

1. The actual internal PS type-name for this object is "EDITFIELD" and *must* be used when creating controls and listing objects. The term "EDITLINE" is used mainly in documentation and the Form Designer.
2. The default property (DEFPROP) is TEXT.
3. The control supports a GLYPH and an IMAGE sub-object.
4. EDITMASK, PASSWORDSTYLE and Windows autocomplete are mutually exclusive modes. PASSWORDSTYLE cannot be enabled while a mask or autocomplete is active. Setting an edit mask disables autocomplete. Password mode prevents changes to EDITMASK, EXITCHARS and VALIDCHARS.
5. Equated constants for use with the EDITLINE object can be found in the PS_EDIT_EQUATES and MSWIN_EDIT_EQUATES insert records.

EDITLINE Properties

The EDITLINE object supports the following properties:

Name	Description
ACCEPTENTER	Specifies whether Return is accepted by the control.
ALWAYSSHOWFOCUSCOLORS	Specifies if the SYSTEM FOCUSSTYLES colors are always applied.
AUTOCOMPLETELIST	Supplies custom strings for Windows autocomplete.
AUTOCOMPLETEMODE	Selects append, suggestion-list or combined autocomplete behaviour.
AUTOCOMPLETESOURCE	Selects the source used by Windows autocomplete.
AUTOSCROLL	Specifies whether text scrolls horizontally as the caret moves beyond the visible width.
AUTOSELECT	Selects the current input automatically when the control receives focus.
CANCOPY	Reports whether the current selection can be copied.
CANCUT	Reports whether the current selection can be cut.
CANPASTE	Reports whether clipboard text can be pasted.
CANUNDO	Reports whether an edit can be undone.
CUEBANNER	Sets prompt text shown when the control has no value.
CUEBANNERSHOWFOCUSED	Specifies whether CUEBANNER remains visible while the control has focus.
EDITMASK	Defines editable positions, literal template characters and the placeholder character.
EDITMASKGROUPSEL	Uses editable mask groups for focus selection and navigation.
EDITSTATEFLAGS	Returns the current availability of edit commands.
ENABLED	Gets or sets the effective enabled state of the edit control.
EXITCHARS	Defines characters that move focus out of the control instead of being inserted.
GETMASKEDCHARONLY	Controls whether TEXT returns only editable mask positions.
HELPFLAGS	Specifies if the object supports HELP or OPTIONS operations.
HIDESELECTION	Specifies whether the selection is hidden when the control loses focus.
LIMITTEXT	Gets or sets the maximum number of characters.
MASKEDTEXT	Gets or sets masked input using editable characters only.
MODIFIED	Gets or sets the Windows edit-control modified flag.
PASSWORDPEEKTIME	Specifies how long the most recently entered password character can remain visible.
PASSWORDSTYLE	Enables password display while retaining the real text value internally.
READONLY	Prevents user edits while preserving selection and copy operations.
SELECTION	Gets or sets the selected range.
SETMASKEDCHARONLY	Controls how TEXT input is interpreted when an edit mask is active.
TABOUTXCEED	Moves focus forward when typing would exceed the text limit.
TEXT	Gets or sets the EDITLINE value. TEXT is the default property.
TEXTALIGN	Sets horizontal text alignment.
TEXTCASE	Selects normal, lowercase or uppercase character handling.
VALIDCHARS	Restricts input to characters in an allow-list.

ACCEPTENTER property

Description

Specifies whether the Enter (Return) key is accepted by the control.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

When FALSE\$, Return follows the form's default-button or focus-navigation behavior. When TRUE\$, EDITLINE consumes Enter; it remains a single-line control and does not create another line.

Example

```
Call Set_Property_Only( CtrlEntID, "ACCEPTENTER", TRUE$ )
```

See Also

Common GUI CHAR event.

ALWAYSSHOWFOCUSCOLORS property

Description

Specifies if the SYSTEM FOCUSSTYLES colors are always applied when focused even if the control has custom FORECOLOR and BACKCOLOR properties.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

N/a.

Example

```
Call Set_Property_Only( CtrlEntID, "ALWAYSSHOWFOCUSCOLORS", TRUE$ )
```

See Also

Common GUI BACKCOLOR property, Common GUI FOCUS property, Common GUI FORECOLOR property, SYSTEM FOCUSSTYLES property, Common GUI GOTFOCUS event, Common GUI LOSTFOCUS event.

AUTOCOMPLETELIST property

Description

Supplies custom strings for Windows autocomplete.

Property Value

An @Fm-delimited list of items for auto-completion. The default is null.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Empty entries are ignored. The list is used when AUTOCOMPLETESOURCE is 0. Replacing the property clears the previous custom list. Autocomplete is unavailable with PASSWORDSTYLE or EDITMASK.

Example

```
Choices = "London" : @Fm : "Londonderry"  
Call Set_Property_Only( CtrlEntID, "AUTOCOMPLETELIST", Choices )
```

See Also

AUTOCOMPLETEMODE property, AUTOCOMPLETESOURCE property, DROPDOWNAUTOCOMplete method.

AUTOCOMPLETEMODE property

Description

Selects append, suggestion-list or combined autocomplete behavior.

Property Value

An integer from 0 through 3. The default is 0.

Value	Description
0	Disabled.
1	Append the best match.
2	Show a suggestion list.
3	Append and show suggestions.

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
Get/Set	Get/Set	No	No	No

Remarks

The control forces this property to 0 when password style or an edit mask is active. Changing the mode attaches or detaches the Windows autocomplete helper.

Example

```
Call Set_Property_Only( CtrlEntID, "AUTOCOMPLETEMODE", PS_EDL_ACM_ALL$ )
```

See Also

AUTOCOMPLETELIST property, AUTOCOMPLETESOURCE property, HELPFLAGS property, PASSWORDSTYLE property, Common GUI OPTIONS event.

AUTOCOMPLETESOURCE property

Description

Selects the source used by Windows autocomplete.

Property Value

An integer from 0 through 5.

Value	Description
0	Custom AUTOCOMPLETelist.
1	File system entries.
2	Directory entries.
3	History.
4	Most-recently-used list.
5	Shell file-system source.

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
Get/Set	Get/Set	No	No	No

Remarks

Source 0 uses AUTOCOMPLETelist. The other values request Windows shell sources. Before the helper has been initialized, a read can report -1 to mean that no source is attached yet.

Example

```
Call Set_Property_Only( CtrlEntID, "AUTOCOMPLETESOURCE", PS_EDL_ACS_DIRLIST$ )
```

See Also

AUTOCOMPLETelist property, AUTOCOMPLETEMODE property.

AUTOSCROLL property

Description

Specifies whether text scrolls horizontally as the caret moves beyond the visible width.

Property Value

A Boolean value. The default is TRUE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get	No	No	No

Remarks

This is a creation-style setting. It can be read at runtime, but runtime updates are not supported.

For more information on this property please refer to the Windows documentation regarding the ES_AUTOHSCROLL style on the Microsoft website.

Example

```
AutoScroll = Get_Property( CtrlEntID, "AUTOSCROLL" )
```

See Also

TEXT property.

AUTOSELECT property

Description

Selects the current input automatically when the control receives focus.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Without an edit mask, all text is selected. With EDITMASK and EDITMASKGROUPSEL enabled, the first editable mask group is selected.

Example

```
Call Set_Property_Only( CtrlEntID, "AUTOSELECT", TRUE$ )
```

See Also

EDITMASK property, EDITMASKGROUPSEL property, SELECTION property.

CANCOPY property

Description

Reports whether the current selection can be copied.

Property Value

A Boolean value.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get	No	No	No

Remarks

TRUE\$ means the selection is non-empty. Use it to enable commands before calling COPY.

Example

```
CanCopy = Get_Property( CtrlEntID, "CANCOPY" )
```

See Also

CANCUT property, CANPASTE property, SELECTION property, COPY method, CUT method, PASTE method, EDITSTATEFLAGS property, CLIPBOARD object.

CANCUT property

Description

Reports whether the current selection can be cut.

Property Value

A Boolean value.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get	No	No	No

Remarks

TRUE\$ requires a selection and an editable, enabled control. Read-only, soft-disabled and hard-disabled states prevent cutting.

Example

```
CanCut = Get_Property( CtrlEntID, "CANCUT" )
```

See Also

CANCOPY property, CANPASTE property, EDITSTATEFLAGS property, ENABLED property, READONLY property, SELECTION property, COPY method, CUT method, PASTE method, CLIPBOARD object.

CANPASTE property

Description

Reports whether clipboard text can be pasted.

Property Value

A Boolean value.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get	No	No	No

Remarks

TRUE\$ requires text on the clipboard and an editable, enabled control. The eventual paste is still filtered by VALIDCHARS and EDITMASK.

Example

```
CanPaste = Get_Property( CtrlEntID, "CANPASTE" )
```

See Also

CANCOPY property, CANCUTE property, EDITMASK property, EDITSTATEFLAGS property, ENABLED property, READONLY property, VALIDCHARS property, COPY method, CUT method, PASTE method, CLIPBOARD object.

CANUNDO property

Description

Reports whether an edit can be undone.

Property Value

A Boolean value.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get	No	No	No

Remarks

The value is the current Windows edit-control undo state.

For more information on this property please refer to the Windows documentation regarding the EM_CANUNDO style on the Microsoft website.

Example

```
CanUndo = Get_Property( CtrlEntID, "CANUNDO" )
```

See Also

UNDO method, RESETUNDO method, EDITSTATEFLAGS property.

CUEBANNER property

Description

Sets prompt text shown when the control has no value.

Property Value

A string. The default is null.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

The cue is display-only and is not returned by TEXT. CUEBANNERSHOWFOCUSED controls whether it remains visible while the control has focus.

For more information on this property please refer to the Windows documentation regarding the EM_SETCUEBANNER message on the Microsoft website.

Example

```
Call Set_Property_Only( CtrlEntID, "CUEBANNER", "Enter account code" )
```

See Also

CUEBANNERSHOWFOCUSED property, TEXT property.

CUEBANNERSHOWFOCUSED property

Description

Specifies whether CUEBANNER remains visible while the control has focus.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

The default behaviour is for a cue-banner to be hidden when the control has focus.

Example

```
Call Set_Property_Only( CtrlEntID, "CUEBANNERSHOWFOCUSED", TRUE$ )
```

See Also

CUEBANNER property, Common GUI GOTFOCUS event.

EDITMASK property

Description

Defines editable positions, literal template characters and the placeholder character.

Property Value

A three-field dynamic array: <1> mask, <2> template and <3> placeholder character. The default placeholder is an underscore.

Value	Description
D	Digit.
d	Digit or space.
+	Plus, minus or space.
C	Alphabetic character.
c	Alphabetic character or space.
A	Alphanumeric character.
a	Alphanumeric character or space.
X	Hexadecimal character.
x	Hexadecimal character or space.
*	Printable character.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Mask and template must have the same length. An invalid or incomplete pair clears the mask and template. In the template, underscore positions are editable and other characters are literals. Setting a mask disables autocomplete. Password mode prevents changing this property.

Example

```
Mask = "DDDDDDDDDD"
Mask<2> = "____-__-____"
Mask<3> = "_"

Call Set_Property_Only( CtrlEntID, "EDITMASK", Mask )
```

See Also

EDITMASKGROUPSEL property, GETMASKEDCHARONLY property, MASKEDTEXT property, SETMASKEDCHARONLY property, Common GUI CHAR event.

EDITMASKGROUPSEL property

Description

Uses editable mask groups for focus selection and navigation.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

This property is meaningful only when EDITMASK is active. With AUTOSELECT, the first editable group is selected when focus arrives.

Example

```
Call Set_Property_Only( CtrlEntID, "EDITMASKGROUPSEL", TRUE$ )
```

See Also

EDITMASK property, AUTOSELECT property, SELECTION property.

EDITSTATEFLAGS property

Description

Returns the current availability of edit commands.

Property Value

An @Fm-delimited Boolean array.

Value	Description
1	Undo.
2	Redo; (always FALSE\$ for EDITLINE objects).
3	Cut.
4	Copy.
5	Paste.
6	Select all.

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
N/a	Get	No	No	No

Remarks

The fields are suitable for enabling edit-menu items. Standard EDITLINE has no redo operation, so field 2 is FALSE\$.

Equates for use with this property can be found the in PS_EQUATES insert record.

Example

```
$Insert PS_Equates

Flags      = Get_Property( CtrlEntID, "EDITSTATEFLAGS" )
CanPaste   = Flags<PS_ESF_CANPASTE$>
```

See Also

EDITSTATECHANGED event, CANCOPY property, CANCUT property, CANPASTE property, CANUNDO property..

ENABLED property

Description

Gets or sets the effective enabled state of the edit control.

Property Value

An integer value. 1 for enabled, 0 for soft-disabled or -1 for hard-disabled. The default is 1.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

A soft-disabled EDITLINE retains the enabled visual style but blocks editing. A hard-disabled control uses the Windows disabled state.

Example

```
Call Set_Property_Only( CtrlEntID, "ENABLED", PS_EDL_SOFTDISABLED$ )
```

See Also

READONLY property, Common GUI ENABLED property.

EXITCHARS property

Description

Defines characters that move focus out of the control instead of being inserted.

Property Value

A string containing the exit characters. The default is null.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

When the user types a listed character, the EDITLINE moves focus forward and does not insert that character. Password mode prevents changing this property.

Example

```
Call Set_Property_Only( CtrlEntID, "EXITCHARS", ";;" )
```

See Also

LIMITTEXT property, TABOUTEXCEED property, VALIDCHARS property, Common GUI CHAR event.

GETMASKEDCHARSONLY property

Description

Controls whether TEXT returns only editable mask positions.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

When TRUE\$ and EDITMASK is active, TEXT omits literal template characters. Placeholder characters in unfilled editable positions remain part of the returned value. MASKEDTEXT temporarily applies this behavior for a single get.

Example

```
Call Set_Property_Only( CtrlEntID, "GETMASKEDCHARSONLY", TRUE$ )  
Value = Get_Property( CtrlEntID, "TEXT" )
```

See Also

EDITMASK property, MASKEDTEXT property, SETMASKEDCHARSONLY property, TEXT property.

HELPFLAGS property

Description

Specifies if the object supports HELP or OPTIONS operations.

Property Value

This property is an @fm-delimited array of boolean flags:

- <1> TRUE\$ if an OPTIONS event is defined for the object, or Autocomplete is enabled.
- <2> TRUE\$ if a HELP event is defined for the object

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
N/a	Get	No	No	Yes

Remarks

This property overrides the standard GUI HELPFLAGS property.

Equates for use with the HELPFLAGS property can be found in the PS_EQUATES insert record.

Example

```
$Insert PS_Equates

// Trigger an OPTIONS event if the control has one.

HelpFlags = Get_Property( CtrlEntID, "HELPFLAGS" )

If HelpFlags<PS_HFL_OPTIONS$> Then
    Call Exec_Method( CtrlEntID, "SHOWOPTIONS" )
End
```

See also

AUTOCOMPLETEMODE property, Common GUI HELPFLAGS property, Common GUI HELP event, Common GUI OPTIONS event.

HIDSELECTION property

Description

Specifies whether the selection is hidden when the control loses focus.

Property Value

A Boolean value. The default is TRUE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get	No	No	No

Remarks

This creation-style setting can be configured in the Form Designer and read at runtime. Runtime updates are not supported.

For more information on this property please refer to the Windows documentation regarding the ES_NOHIDSESEL style on the Microsoft website.

Example

```
HideSelection = Get_Property( CtrlEntID, "HIDSELECTION" )
```

See Also

SELECTION property, Common GUI GOTFOCUS event, Common GUI LOSTFOCUS event.

LIMITTEXT property

Description

Gets or sets the maximum number of characters allowed for the control.

Property Value

An unsigned integer.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

The limit is measured by the edit control after any character-map translation.

The LIMIT property is a deprecated runtime alias for this property. It is still supported for backwards compatibility.

For more information on this property please refer to the Windows documentation regarding the EM_GETLIMITTEXT and EM_SETLIMITTEXT messages on the Microsoft website.

Example

```
Call Set_Property_Only( CtrlEntID, "LIMITTEXT", 20 )
```

See Also

EXITCHARS property, TABOUTEXCEED property, TEXT property.

MASKEDTEXT property

Description

Gets or sets masked input using editable characters only.

Property Value

A string value.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get/Set	No	No	No

Remarks

This property is meaningful only when EDITMASK is active.

For this one operation, MASKEDTEXT behaves like TEXT with both GETMASKEDCHARONLY and SETMASKEDCHARONLY enabled. Literal template characters are supplied by the mask rather than the caller.

Example

```
Call Set_Property_Only( CtrlEntID, "MASKEDTEXT", "1234567890" )
Digits = Get_Property( CtrlEntID, "MASKEDTEXT" )
```

See Also

EDITMASK property, GETMASKEDCHARONLY property, SETMASKEDCHARONLY property, TEXT property.

MODIFIED property

Description

Gets or sets the control's modified flag.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
N/a	Get/Set	No	No	No

Remarks

The flag can be reset after saving. It is independent of application validation and does not compare TEXT with a stored database value.

For more information on this property please refer to the Windows documentation regarding the EM_GETMODIFY and EM_SETMODIFY messages on the Microsoft website.

Example

```
Call Set_Property_Only( CtrlEntID, "MODIFIED", FALSE$ )
```

See Also

TEXT property, PASTE method, CHAR event, CHANGED event.

PASSWORDPEEKTIME property

Description

Specifies how long the most recently entered password character can remain visible.

Property Value

An integer value specifying the number of milliseconds. The default is 0.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

The property applies only while PASSWORDSTYLE is active.

Example

```
Call Set_Property_Only( CtrlEntID, "PASSWORDPEEKTIME", 750 )
```

See Also

PASSWORDSTYLE property, TEXT property.

PASSWORDSTYLE property

Description

Enables "password character" display while retaining the real text value internally.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

Enabling password style is refused while EDITMASK or autocomplete is active. In password mode, TEXT and CHANGED expose the real value, while the window displays "password characters". Disabling the property restores the real value to the control.

Example

```
Call Set_Property_Only( CtrlEntID, "PASSWORDSTYLE", TRUE$ )
```

See Also

EDITMASK property, PASSWORDPEEKTIME property, TEXT property, CHANGED event.

READONLY property

Description

Prevents user edits while preserving selection and copy operations.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

READONLY differs from ENABLED: users can still focus, select and copy text. The OPTIONS availability flag is suppressed while the EDITLINE is read-only.

For more information on this property please refer to the Windows documentation regarding the ES_READONLY style and the EM_SETREADONLY message on the Microsoft website.

Example

```
Call Set_Property_Only( CtrlEntID, "READONLY", TRUE$ )
```

See Also

CANCUT property, CANPASTE property, ENABLED property, Common GUI OPTIONS event.

SELECTION property

Description

Gets or sets the selected range of characters.

Property Value

A two-field @fm delimited dynamic array:

<1> 1-based start position
<2> Selection length

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
N/a	Get/Set	No	No	No

Remarks

Setting a start position of zero is treated as position 1. After setting the range, the control scrolls the caret into view. A zero length places the caret without selecting characters.

For more information on this property please refer to the Windows documentation regarding the EM_GETSEL and EM_SETSEL messages on the Microsoft website.

Example

```
Selection      = 1
Selection<2>  = 4

Call Set_Property_Only( CtrlEntID, "SELECTION", Selection )
```

See Also

AUTOSELECT property, SCROLLTOCARET method, SELECTALL method.

SETMASKEDCHARONLY property

Description

Controls how TEXT input is interpreted when an edit mask is active.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

When TRUE\$, a TEXT assignment supplies editable characters only and the EDITLINE inserts template literals. When FALSE\$, the caller supplies text in the full masked layout.

Example

```
Call Set_Property_Only( CtrlEntID, "SETMASKEDCHARONLY", TRUE$ )
Call Set_Property_Only( CtrlEntID, "TEXT", "1234567890" )
```

See Also

EDITMASK property, GETMASKEDCHARONLY property, MASKEDTEXT property, TEXT property.

TABOUTEXCEED property

Description

Moves focus forward when typing would exceed the text limit.

Property Value

A Boolean value. The default is FALSE\$.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

The behaviour applies when the current text is at the configured limit and no selected characters can be replaced. It provides a data-entry flow option; it does not increase LIMITTEXT.

Example

```
Call Set_Property_Only( CtrlEntID, "TABOUTEXCEED", TRUE$ )
```

See Also

EXITCHARS property, LIMITTEXT property.

TEXT property

Description

Gets or sets the EDITLINE value. TEXT is the default property.

Property Value

A string value.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

In password mode, TEXT exchanges the internally retained real value rather than the "password characters" shown by Windows. With EDITMASK, GETMASKEDCHARONLY and SETMASKEDCHARONLY control whether literal template characters participate. Character-map (CHARMAP) translation is applied by the property interface.

Example

```
Call Set_Property_Only( CtrlEntID, "TEXT", "A123" )  
Value = Get_Property( CtrlEntID, "TEXT" )
```

See Also

EDITMASK property, MASKEDTEXT property, PASSWORDSTYLE property, CHANGED event.

TEXTALIGN property

Description

Sets horizontal text alignment.

Property Value

Can be one of the following horizontal text-align values:

Value	Description
0	Align Left (the default).
1	Align Center.
2	Align Right.

Or a character value (set only):

Value	Description
'L' or 'l'	Align Left (the default).
'C' or 'c'	Align Center.
'R' or 'r'	Align Right.

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
Get/Set	Get/Set	No	No	No

Remarks

Equates for the horizontal text-align values can be found in the PS_EQUATES insert record.

For more information on this property please refer to the Windows documentation regarding the ES_RIGHT and ES_CENTER styles on the Microsoft website.

Example

```
$Insert PS_Equates  
  
Call Set_Property_Only( CtrlEntID, "TEXTALIGN", PS_HTA_RIGHT$ )
```

See Also

CANCUT property, CANPASTE property, ENABLED property, Common GUI OPTIONS event.

TEXTCASE property

Description

Selects normal, lowercase or uppercase character handling.

Property Value

An integer value:

Value	Description
0	Mixed/normal case (The default).
1	Lowercase.
2	Uppercase.

Property Traits

Development	Runtime	Indexed	Scaled	Synthetic
Get/Set	Get/Set	No	No	No

Remarks

This property affects user input and displayed text; application code should not treat it as locale-independent data normalization.

For more information on this property please refer to the Windows documentation regarding the ES_LOWERCASE and ES_UPPERCASE styles on the Microsoft website.

Example

```
Call Set_Property_Only( CtrlEntID, "TEXTCASE", PS_EDX_TEXTCASE_UPPER$ )
```

See Also

TEXT property, VALIDCHARS property.

VALIDCHARS property

Description

Restricts input to characters in an allow-list.

Property Value

A string containing the valid characters. The default is null, which means unrestricted.

Property Traits

<i>Development</i>	<i>Runtime</i>	<i>Indexed</i>	<i>Scaled</i>	<i>Synthetic</i>
Get/Set	Get/Set	No	No	No

Remarks

The restriction is combined with EDITMASK when both are present. Setting the property revalidates the current text. Paste processing filters characters that do not satisfy the active restrictions. Password mode prevents changing this property.

Example

```
Call Set_Property_Only( CtrlEntID, "VALIDCHARS", "0123456789" )
```

See Also

EDITMASK property, EXITCHARS property, PASTE method, TEXT property.

EDITLINE Methods

The EDITLINE object supports the following methods:

Name	Description
APPEND	Appends text at the end of the current value.
COPY	Copies the current selection to the clipboard.
CUT	Cuts the current selection to the clipboard.
DELETESELECTION	Deletes the current selection.
DROPDOWNAUTOCOMPLETE	Requests display of the autocomplete suggestion list.
INSERT	Inserts text at a selected position.
PASTE	Pastes clipboard text at the current selection.
RESETUNDO	Clears the edit control's undo history.
SCROLLTOCARET	Scrolls the control so the caret is visible.
SELECTALL	Selects all text.
SHOWBALLOONTIP	Displays a Windows edit-control balloon tip.
UNDO	Undoes the last edit-control operation.

APPEND Method

Description

Appends text at the end of the current value.

Syntax

```
Success = Exec_Method( CtrlEntID, "APPEND", Text )
```

Parameters

Name	Required	Description
Text	Yes	Text to append.

Returns

TRUE\$ when Text is supplied; otherwise FALSE\$.

Remarks

Character-map translation is applied before insertion.

Example

```
Success = Exec_Method( CtrlEntID, "APPEND", " Ltd" )
```

See Also

TEXT property, INSERT method.

COPY Method

Description

Copies the current selection to the clipboard.

Syntax

```
Success = Exec_Method( CtrlEntID, "COPY" )
```

Parameters

N/a.

Returns

TRUE\$ after the copy message is sent.

Remarks

The return value does not confirm that a non-empty selection was copied. Read the CANCOPY property before calling when command availability matters.

For more information on this method please refer to the Windows documentation regarding the WM_COPY message on the Microsoft website.

Example

```
If Get_Property( CtrlEntID, "CANCOPY" ) Then  
    Call Exec_Method( CtrlEntID, "COPY" )  
End
```

See Also

CANCOPY property, SELECTION property, CLIPBOARD object.

CUT Method

Description

Cuts the current selection to the clipboard.

Syntax

```
Success = Exec_Method( CtrlEntID, "CUT" )
```

Parameters

N/a.

Returns

TRUE\$ after the cut message is sent.

Remarks

The return does not prove that text changed. Read the CANCUT property to account for selection, read-only and disabled state.

For more information on this method please refer to the Windows documentation regarding the WM_CUT message on the Microsoft website.

Example

```
If Get_Property( CtrlEntID, "CANCUT" ) Then  
    Call Exec_Method( CtrlEntID, "CUT" )  
End
```

See Also

CANCUT property, COPY method, DELETEDSELECTION method, CLIPBOARD object.

DELETEDELECTION Method

Description

Deletes the current selection.

Syntax

```
Success = Exec_Method( CtrlEntID, "DELETEDELECTION" )
```

Parameters

N/a.

Returns

TRUE\$ after the delete message is sent.

Remarks

A read-only or empty selection can leave the text unchanged even though the method returns TRUE\$. Read the CANCEL property to account for selection, read-only and disabled state.

For more information on this method please refer to the Windows documentation regarding the WM_CLEAR message on the Microsoft website.

Example

```
If Get_Property( CtrlEntID, "CANCEL" ) Then  
    Call Exec_Method( CtrlEntID, "DELETEDELECTION" )  
End
```

See Also

CANCEL property, SELECTION property.

DROPDOWNAUTOCOMPLETE Method

Description

Requests display of the autocomplete suggestion list.

Syntax

```
Success = Exec_Method( CtrlEntID, "DROPDOWNAUTOCOMPLETE" )
```

Parameters

N/a.

Returns

TRUE\$ when autocomplete mode is enabled and the request is posted; otherwise FALSE\$.

Remarks

The request posts a Down-arrow keystroke to the control.

Example

```
Success = Exec_Method( CtrlEntID, "DROPDOWNAUTOCOMPLETE" )
```

See Also

AUTOCOMPLETELIST property, AUTOCOMPLETEMODE property.

INSERT Method

Description

Inserts text at a selected position.

Syntax

```
PreviousPos = Exec_Method( CtrlEntID, "INSERT", InsertPos, Text )
```

Parameters

Name	Required	Description
InsertPos	Yes	0 or null for the current selection, a negative value for the end, or a positive 1-based position.
Text	Yes	Text to insert.

Returns

The 1-based selection start that was active before insertion.

Remarks

Character-map translation is applied before insertion.

Example

```
PreviousPos = Exec_Method( CtrlEntID, "INSERT", -1, "!" )
```

See Also

SELECTION property, TEXT property, APPEND method.

PASTE Method

Description

Pastes clipboard text at the current selection.

Syntax

```
Success = Exec_Method( CtrlEntID, "PASTE" )
```

Parameters

N/a.

Returns

TRUE\$ after the paste message is sent.

Remarks

The return does not prove that text changed. Read CANPASTE first. EDITMASK and VALIDCHARS filter pasted input.

For more information on this method please refer to the Windows documentation regarding the WM_PASTE message on the Microsoft website.

Example

```
If Get_Property( CtrlEntID, "CANPASTE" ) Then  
    Call Exec_Method( CtrlEntID, "PASTE" )  
End
```

See Also

CANPASTE property, EDITMASK property, VALIDCHARS property, CLIPBOARD object.

RESETUNDO Method

Description

Clears the edit control's undo history.

Syntax

```
Success = Exec_Method( CtrlEntID, "RESETUNDO" )
```

Parameters

N/a.

Returns

TRUE\$ after the undo buffer is cleared.

Remarks

After this call, CANUNDO is normally FALSE\$ until another editable operation occurs.

For more information on this method please refer to the Windows documentation regarding the EM_EMPTYUNDOBUFFER message on the Microsoft website.

Example

```
If Get_Property( CtrlEntID, "CANUNDO" ) Then  
    Call Exec_Method( CtrlEntID, "RESETUNDO" )  
End
```

See Also

CANUNDO property, UNDO method.

SCROLLTOCARET Method

Description

Scrolls the control so the caret is visible.

Syntax

```
Success = Exec_Method( CtrlEntID, "SCROLLTOCARET" )
```

Parameters

N/a.

Returns

TRUE\$ after the scroll request.

Remarks

SELECTION already scrolls the caret after setting a range; use this method after other caret-changing operations when necessary.

For more information on this method please refer to the Windows documentation regarding the EM_SCROLLCARET message on the Microsoft website.

Example

```
Call Exec_Method( CtrlEntID, "SCROLLTOCARET" )
```

See Also

AUTOSCROLL property, SELECTION property.

SELECTALL Method

Description

Selects all text.

Syntax

```
Success = Exec_Method( CtrlEntID, "SELECTALL" )
```

Parameters

N/a.

Returns

TRUE\$ after the text is selected.

Remarks

For more information on this method please refer to the Windows documentation regarding the EM_SETSEL message on the Microsoft website.

Example

```
Call Exec_Method( CtrlEntID, "SELECTALL" )
```

See Also

AUTOSELECT property, SELECTION property.

SHOWBALLOONTIP Method

Description

Displays a Windows edit-control balloon tip.

Syntax

```
Success = Exec_Method( CtrlEntID, CtrlEntID, "SHOWBALLOONTIP", Text, |  
                        Title, IconID )
```

Parameters

Name	Required	Description
Text	Yes	Balloon message text.
Title	No	Optional title.
IconID	No	Numeric value specifying the icon to display: 0 No icon 1 Information 2 Warning 3 Error 4 Large Information 5 Large Warning 6 Large Error

Returns

TRUE\$ if the control accepts the request; otherwise FALSE\$.

Remarks

Equates for use with this method can be found in the PS_EDIT_EQUATES insert record.

For more information on this method please refer to the Windows documentation regarding the EM_SHOWBALLOONTIP message on the Microsoft website.

Example

```
Success = Exec_Method( CtrlEntID, "SHOWBALLOONTIP",  
                        "Enter a valid account code", "Input", |  
                        PS_EDX_TTI_WARNING$ )
```

See Also

Common GUI TOOLTIP property.

UNDO Method

Description

Undoes the last edit-control operation.

Syntax

```
Success = Exec_Method( CtrlEntID, "UNDO" )
```

Parameters

N/a.

Returns

TRUE\$ after the undo operation succeeded, or FALSE\$ otherwise.

Remarks

Read the CANUNDO or EDITSTATEFLAGS properties before enabling an Undo command. RESETUNDO clears the available history.

For more information on this method please refer to the Windows documentation regarding the EM_UNDO message on the Microsoft website.

Example

```
If Get_Property( CtrlEntID, "CANUNDO" ) Then  
    Call Exec_Method( CtrlEntID, "UNDO" )  
End
```

See Also

CANUNDO property, EDITSTATEFLAGS property, RESETUNDO method, EDITSTATECHANGED event.

EDITLINE Events

The EDITLINE object supports the following events:

Name	Description
CHANGED	Reports a change to the control's value.
EDITSTATECHANGED	Reports a change in edit-command availability.

CHANGED event

Description

Reports a change to the control's value.

Syntax

```
bForward = CHANGED( CtrlEntID, CtrlClassID, NewData )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the EDITLINE object receiving the event.
CtrlClassID	Type of object receiving the event (always "EDITLINE").
NewData	The contents of the EDITLINE <i>at the time the CHANGE event was triggered</i> .

Returns

TRUE\$ or FALSE\$. If FALSE\$, the program execution returns to the calling procedure. If TRUE\$, the event processing goes to the next level.

Remarks

There is no system-level promoted event handler for the CHANGED event.

CHANGED is the default EDITLINE event. In password mode, NewData contains the retained real text rather than the displayed password characters. For masked input, use TEXT or MASKEDTEXT after the event when the required literal/character-only form matters.

For more information on this topic please refer to the Windows EN_CHANGE notification message on the Microsoft website.

Example

```
Function CHANGED( CtrlEntID, CtrlClassID, NewData )

    // If the control contains data then enable the OK button,
    // otherwise disable it.

    // Get the TEXT property to ensure we have the _actual_ current
    // data (events are normally handled asynchronously)
    CurrData = Get_Property( CtrlEntID, "TEXT" )

    Enabled = ( BLen( CurrData ) > 0 )

    Call Set_Property_Only( @Window : "BTN_OK", "ENABLED", Enabled )

Return 1
```

See Also

MODIFIED property, PASSWORDSTYLE property, TEXT property, CHAR event.

EDITSTATECHANGED event

Description

Reports a change in edit-command availability.

Syntax

```
bForward = EDITSTATECHANGED( CtrlEntID, CtrlClassID, EditState )
```

Parameters

Name	Description
CtrlEntID	Fully qualified name of the EDITLINE object receiving the event.
CtrlClassID	Type of object receiving the event (always "EDITLINE").
EditState	An @Vm-delimited array of Boolean edit-state flags: <0,1> CanUndo <0,2> CanRedo <0,3> CanCut <0,4> CanCopy <0,5> CanPaste <0,6> CanSelectAll

Returns

TRUE\$ or FALSE\$. If FALSE\$, the program execution returns to the calling procedure. If TRUE\$, the event processing goes to the next level.

Remarks

The system-level EDITSTATECHANGED event handler performs the following tasks:

- Sets the state of the appropriate Edit menu items in the IDE if running in an IDE context.

The edit is raised if a visible control when the state differs from the last reported state.

The EDITSTATEFLAGS property returns the same fields with @Fm delimiters. The CanRedo flag is always FALSE\$ for the EDITLINE control.

Equates for use with the EDITSTATECHANGED event can be found in the PS_EQUATES insert record.

Example

```
Function EDITSTATECHANGED( CtrlEntID, CtrlClassID, EditState )

    $Insert PS_Equates
    $Insert Logical

    // Set the ENABLED state of some clipboard buttons to match
    // the passed EditState

    CanCut    = EditState<PS_ESF_CANCUT$>
    CanCopy   = EditState<PS_ESF_CANCOPY$>
    CanPaste  = EditState<PS_ESF_CANPASTE$>

    Call Set_Property_Only( @Window : ".BTN_CUT", "ENABLED", CanCut )
    Call Set_Property_Only( @Window : ".BTN_COPY", "ENABLED", CanCopy )
    Call Set_Property_Only( @Window : ".BTN_PASTE", "ENABLED", CanPaste )

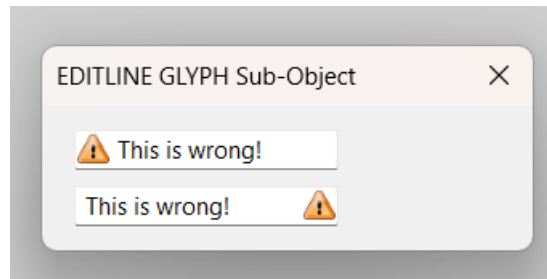
    Return 1
```

See Also

CANCOPY property, CANCUT property, CANPASTE property, CANUNDO property, EDITSTATEFLAGS property.

EDITLINE GLYPH Sub-Object

The EDITLINE object supports a GLYPH sub-object which allows an image to be displayed to the right or left of the EDITLINE in its non-client area. This can be useful for displaying warning indications such as “data is required” and so on.



The GLYPH sub-object supports all the standard Image Object API properties and methods with the following exceptions:

- **ALIGN:** Only Left and Right alignment values are supported, and all vertical alignments are ignored.

Example

```
// Example: set a glyph image and align it to the right if data is
// missing from a LOSTFOCUS event

$Insert PS_Equates

Text = Get_Property( CtrlEntID, "TEXT" )
If BLen( Text ) Then
    // Remove the warning icon
    GlyphFile = ""
End Else
    GlyphFile = ".\images\warning.png"
    GlyphAlign = PS_IA_TOPRIGHT$
End

GlyphID = CtrlEntID : ".GLYPH"

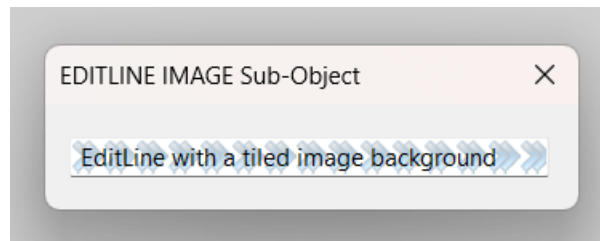
Call Set_Property_Only( GlyphID, "FILENAMES", GlyphFile )
Call Set_Property_Only( GlyphID, "ALIGN", GlyphAlign )
```

See Also

IMAGE sub-object, Image API documentation.

EDITLINE IMAGE Sub-Object

The EDITLINE object supports an IMAGE sub-object which allows an image to be displayed as the background to the control.



The IMAGE sub-object supports all the standard Image Object API properties and methods.

Example

```
// Example: set a tiled translucent background  
  
$Insert PS_Equates  
  
ImageID = CtrlEntID : ".IMAGE"  
  
FileName = ".\images\arrows.png"  
  
Call Set_Property_Only( ImageID , "FILENAMES", FileName )  
Call Set_Property_Only( ImageID , "STYLE", PS_IS_TILE$ )  
Call Set_Property_Only( ImageID , "TRANSLUCENCY", 70 )
```

See Also

GLYPH sub-object, Image API documentation.