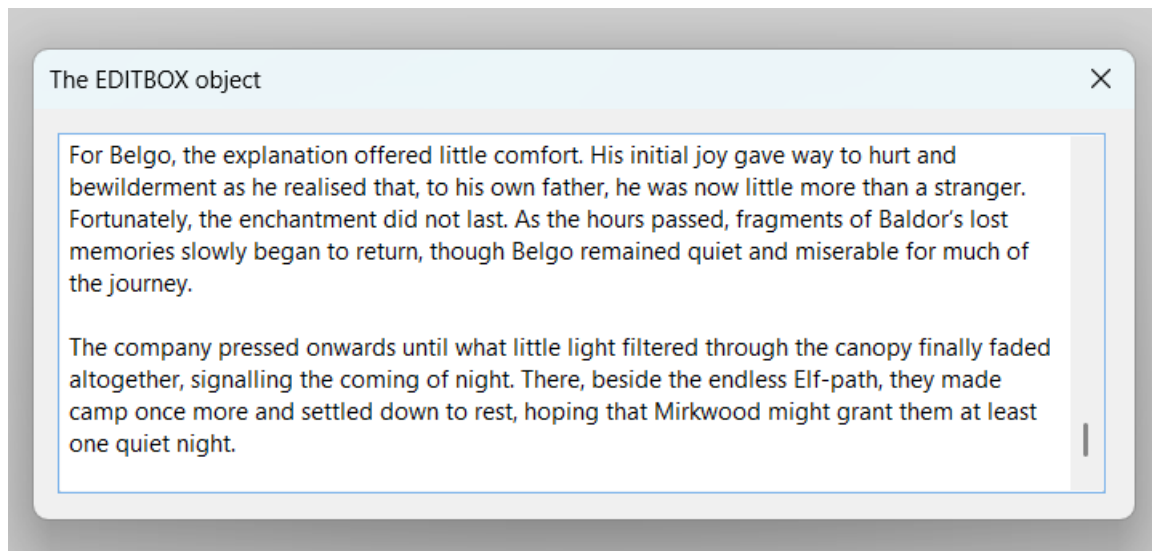


EDITBOX object

The EDITBOX is the basic multiline OpenInsight text-entry control. It supports line-oriented input, word wrapping, vertical scrolling, tab stops and optional automatic indentation.



Developer Notes

1. The default property (DEFPROP) is TEXTVAL.
2. The control supports an IMAGE sub-object.
3. 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.
4. Equated constants for use with the EDITBOX object can be found in the PS_EDIT_EQUATES and MSWIN_EDIT_EQUATES insert records.

EDITBOX Properties

The EDITBOX object supports the following properties:

Name	Description
ACCEPTTAB	Specifies whether Tab is inserted or moves focus.
ALWAYSSHOWFOCUSCOLORS	Specifies if the SYSTEM FOCUSSTYLES colors are always applied.
AUTOINDENT	Duplicates the previous line's leading spaces and tabs when a new line is inserted.
AUTOSCROLL	Controls automatic vertical scrolling as new lines are entered.
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.
CARETPOS	Gets or sets the caret's line-relative position.
EDITSTATEFLAGS	Returns the current availability of edit commands.
ENABLED	Gets or sets the effective enabled state of the control.
HELPFLAGS	Specifies if the object supports HELP or OPTIONS operations.
HIDeselection	Specifies whether the selection is hidden when the control loses focus.
LIMIT	Gets or sets the maximum number of characters at runtime.
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.
READONLY	Prevents user edits while preserving selection and copy operations.
SELECTION	Gets or sets the selected range.
SOFTDISABLED	Configures the design-time soft-disabled state.
TABOUTEXCEED	Moves focus forward when typing would exceed the text limit.
TABSTOPS	Gets or sets tab-stop positions for displayed tab characters.
TEXT	Gets or sets the native EDITBOX text.
TEXTALIGN	Sets horizontal text alignment.
TEXTCASE	Selects normal, lowercase or uppercase character handling.
TEXTVAL	Gets or sets multiline text using Basic+ line delimiters. TEXTVAL is the default property.
TEXTVALVM	Selects @Vm rather than @Tm for TEXTVAL getter output.
WORDWRAP	Controls whether long lines wrap within the visible width.

ACCEPTTAB property

Description

Specifies whether the Tab key is accepted by the control and inserted into the text or if it moves the 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

When FALSE\$, Tab moves focus and Ctrl+Tab inserts a tab. When TRUE\$, Tab inserts a tab and Ctrl+Tab moves focus. A read-only EDITBOX always moves focus. This escape rule prevents an EDITBOX that accepts tabs from trapping keyboard focus.

Example

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

See Also

READONLY property, TABSTOPS property, 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.

AUTOINDENT property

Description

Copies the previous line's leading spaces and tabs when a new line is inserted.

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, "AUTOINDENT", TRUE$ )
```

See Also

TEXT property, TEXTVAL property, Common GUI CHAR event.

AUTOSCROLL property

Description

Controls automatic vertical scrolling as new lines are entered.

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_AUTOVSCROLL style on the Microsoft website.

Example

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

See Also

WORDWRAP property, Common GUI SCROLLBARS 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.

CARETPOS property

Description

Gets or sets the caret's line-relative position.

Property Value

This property is an @fm-delimited array integers:

- <1> The 1-based column index.
- <2> The 1-based line index.

Property Traits

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

Remarks

This property does not validate positions beyond the available text.

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

Example

```
// Example: Set the caret to the first character on the third line  
  
Pos      = ""  
Pos<1>   = 1  
Pos<2>   = 3  
  
Call Set_Property_Only( CtrlEntID, "CARETPOS", Pos )
```

See Also

SELECTION property, TEXT property, TEXTVAL property, SCROLLTOCARET method.

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 EDITBOX 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 EDITBOX 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 EDITBOX 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 EDITBOX 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.

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.

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

TABOUTEXCEED property, TEXT property, TEXTVAL 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.

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 EDITBOX 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>  = 10

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

See Also

CARETPOS property, SCROLLTOCARET method, SELECTALL method.

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.

TABSTOPS property

Description

Gets or sets tab-stop positions for displayed tab characters.

Property Value

An @Fm-delimited list of nonzero unsigned integers. With no stored list, the getter returns the platform-dependent default derived from 32 dialog units.

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

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

Example

```
DefaultStops = Get_Property( CtrlEntID, "TABSTOPS" )  
  
Stops = "64" : @Fm : "128"  
Call Set_Property_Only( CtrlEntID, "TABSTOPS", Stops )
```

See Also

ACCEPTTAB property, TEXT property, TEXTVAL property.

TEXT property

Description

Gets or sets the EDITBOX text using CRLF characters for line delimiters.

Property Value

A string containing Windows CRLF line endings.

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

Use TEXT when native character positions or literal CRLF sequences matter. TEXTVAL is the default property and converts line endings to Basic+ delimiters.

Example

```
$Insert RTI_Text_Equates  
  
NewText = "First line" : CRLF$ : "Second line"  
Call Set_Property_Only( CtrlEntID, "TEXT", NewText )
```

See Also

TEXTVAL property, TEXTVALVM 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.

TEXTVAL property

Description

Gets or sets multiline text using Basic+ line delimiters. TEXTVAL is the default property.

Property Value

A string – multiple lines are delimited by @Tm by default, or by @Vm when TEXTVALVM is TRUE\$.

Property Traits

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

Remarks

On set, both @Vm and @Svm are converted to @Tm, then every @Tm becomes CRLF before native TEXT is set. On get, each CRLF becomes the delimiter selected by TEXTVALVM. The conversion therefore normalizes @Vm, @Svm and @Tm input to line breaks and does not preserve their original delimiter levels.

Example

```
$Insert RTI_Text_Equates  
  
NewText = "First line" : @Tm : "Second line"  
Call Set_Property_Only( CtrlEntID, "TEXT", NewText )
```

See Also

TEXT property, TEXTVALVM property.

TEXTVALVM property

Description

Selects @Vm rather than @Tm for TEXTVAL getter output.

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

On set, both @Vm and @Svm are converted to @Tm, then every @Tm becomes CRLF before native TEXT is set. On get, each CRLF becomes the delimiter selected by TEXTVALVM. The conversion therefore normalizes @Vm, @Svm and @Tm input to line breaks and does not preserve their original delimiter levels.

Example

```
$Insert RTI_Text_Equates  
  
NewText = "First line" : @Tm : "Second line"  
Call Set_Property_Only( CtrlEntID, "TEXT", NewText )
```

See Also

TEXT property, TEXTVALVM property.

WORDWRAP property

Description

Controls whether long lines wrap within the visible width.

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	No	No	No

Remarks

WORDWRAP is a creation-style setting. It can be configured in the Form Designer and read at runtime.

Example

```
WordWrap = Get_Property( CtrlEntID, "WORDWRAP" )
```

See Also

AUTOSCROLL property, Common GUI SCROLLBARS property.

EDITBOX Methods

The EDITBOX 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.
DELETSELECTION	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

This method operates on native TEXT, not TEXTVAL delimiter conversion. Construct CRLF explicitly when appending lines.

Character-map translation is applied before insertion.

Example

```
$Insert RTI_Text_Equates  
  
Success = Exec_Method( CtrlEntID, "APPEND", CRLF$ : "Next line" )
```

See Also

TEXT property, TEXTVAL 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.

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. Use CRLF for a line break.

Returns

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

Remarks

This method operates on native TEXT, not TEXTVAL delimiter conversion. Construct CRLF explicitly when inserting lines.

Character-map translation is applied before insertion.

Example

```
$Insert RTI_Text_Equates  
  
InsertPos = Exec_Method( CtrlEntID, "INSERT", CRLF$ : "New line" : CRLF$ )
```

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.

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, 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, CARETPOS 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

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", |  
                        "Add more detail!", "Description", |  
                        PS_EDX_TTI_INFO$ )
```

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.

EDITBOX Events

The EDITBOX 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 EDITBOX object receiving the event.
CtrlClassID	Type of object receiving the event (always "EDITBOX").
NewData	The contents of the EDITBOX in TEXT format <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 EDITBOX event.

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 )  
  
    // Get the TEXTVAL property to ensure we have the _actual_ current  
    // data (events are normally handled asynchronously)  
    CurrText = Get_Property( CtrlEntID, "TEXTVAL" )  
  
    Return 1
```

See Also

TEXT property, TEXTVAL 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 EDITBOX object receiving the event.
CtrlClassID	Type of object receiving the event (always "EDITBOX").
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 EDITBOX 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 )

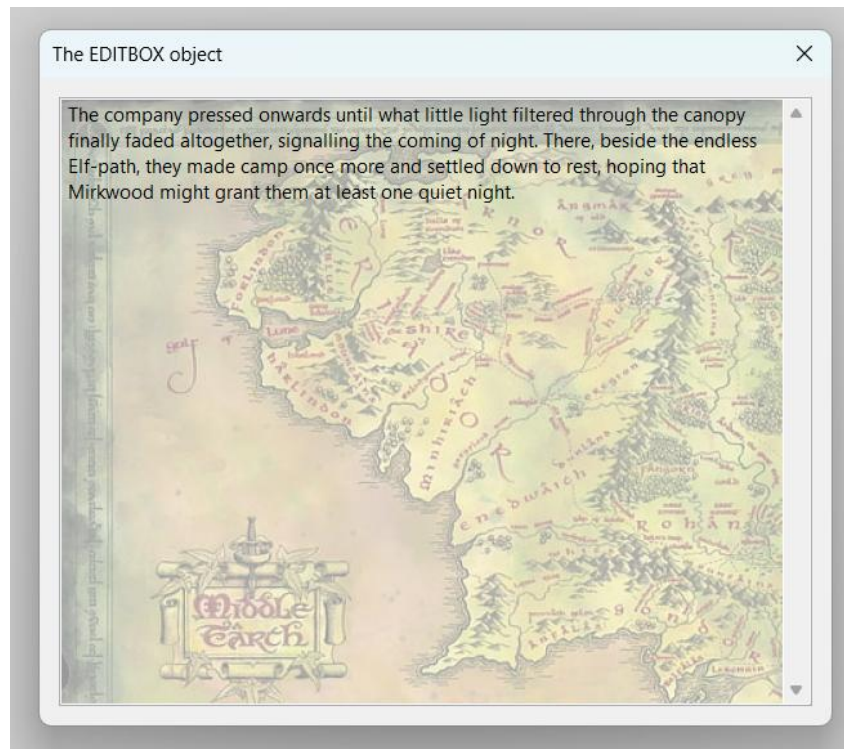
    Return 1
```

See Also

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

EDITBOX IMAGE Sub-Object

The EDITBOX 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 translucent background

$Insert PS_Equates

ImageID = CtrlEntID : ".IMAGE"

FileName = ".\images\map.png"

Call Set_Property_Only( ImageID , "FILENAMES", FileName )
Call Set_Property_Only( ImageID , "STYLE", PS_IS_CLIP$ )
Call Set_Property_Only( ImageID , "TRANSLUCENCY", 60 )
```

See Also

Image API documentation.