Freelance Graphics LotusScript A-Z

Freelance Graphics LotusScript Classes A-Z

coordinates (defined)

The Freelance Graphics coordinate system has its origin (0,0) at the bottom left corner of the page. Coordinates are measured in twips (1/1440 of an inch, or 1/567 of a centimeter). In LotusScript syntax, the horizontal coordinate is usually referred to as *x*, and the vertical coordinate as *y*.

Freelance Graphics: Application class
Controls a Freelance Graphics session.

Base classes
BaseObject

Contained by

Class	Property
Document	Application

Usage
You can use the CurrentApplication predefined variable to reference the properties and methods of the current Application object.

Freelance Graphics: Application class members

Properties

ActiveDocument
ActiveDocWindow
Application AS Application class
ApplicationWindow
Colors AS Colors class
CurrentPrinter
DefaultFilePath
Documents AS Documents class
FontUnit
FullName
Interactive
Location
PageSelection AS PageSelection class
Parent
Path
Preferences AS Preferences class
Selection AS Selection class
UnitOfMeasure
Visible

Methods

CloseWindow
GetEnum
NearestColorFromRGB
NewDocument
OpenDocument
OpenDocumentFromInternet
OpenDocumentFromNotes
Quit
SetInternetOptions

Functions

None

Events

None

Freelance Graphics: ApplicationWindow class members

Properties

Height
Width

Methods

Cascade
Close
GotoNotes
Maximize
Minimize
Restore
Tile

Functions

None

Events

None

Freelance Graphics: ApplicationWindow class

The application's main window.

Base classes

BaseObject

Contained by

None

Usage

You can use the CurrentApplicationWindow predefined variable to reference the properties and methods of the currently selected ApplicationWindow object.

Freelance Graphics: Background class
Properties (color, fill pattern, etc.) of an object's background.

Base classes
BaseObject

Contained by

Class	Property
DrawObject	Background

Freelance Graphics: Background class members

Properties

BackColor
Color AS Color class
Pattern

Methods

RevertToStyle

Functions

None

Events

None

Freelance Graphics: BaseObject class

Abstract class used as the base class for all Freelance Graphics objects - no instances of this class ever exist.

Base classes

None

Contained by

All Freelance Graphics LotusScript classes. See Freelance Graphics LotusScript Classes A-Z.

Freelance Graphics: BaseObject class members

The BaseObject class is an abstract class used as the base class for all Freelance Graphics objects - no instances of this class ever exist. It contains the following properties.

Properties

- Application
- Description
- IsSelectable
- IsValid
- Name
- Parent
- VersionID

Methods

None

Functions

None

Events

None

Freelance Graphics: Border class

Properties of an object's edges.

Base classes

BaseObject

Contained by

Class	Property
DrawObject	Border

Freelance Graphics: Border class members

Properties

Color AS Color class
Pattern
Width

Methods

RevertToStyle

Functions

None

Events

None

Freelance Graphics: BulletProperties class members

Properties

Color AS Color class
ShadowColor
ShadowDepth
ShadowDirection
Size
StartNumber
Style

Methods

None

Functions

None

Events

None

Freelance Graphics: BulletProperties class

Level-specific bullet properties.

Base classes

BaseObject

Contained by

Class	Property
TextBlock	BulletProperties

Freelance Graphics: Chart class members

Properties

- Application
- Background
- Border
- BuildBullets
- Chart
- Delay
- Description
- ExeName
- Height
- IsChart
- IsDraggable
- IsGroup
- IsImage
- IsMedia
- IsOleObj
- IsPlacementBlock
- IsSelectable
- IsTable
- IsText
- IsValid
- Left
- LineLead
- LineStyle
- Media
- Name
- OleObj
- Page
- Parent
- PlacementBlock
- PlayPriority
- Priority
- Table
- TextBlock
- Top
- TransitionEffect
- VersionID
- WaitForClick
- Width

Methods

- AddPoint
- ConvertTo
- Copy
- Cut
- Flip
- GetObjectData
- Item
- Move
- PutIntoPlacementBlock
- Remove
- Rotate
- SetObjectData
- Stretch
- Ungroup

Functions

None

Events

None

Freelance Graphics: Chart class

A chart of any type. The Freelance Graphics Chart class is derived from the DrawObject class and contains all methods and properties of the ChartBase class. You create a chart using the CreateChart method of the Page class.

Note For more information about the LotusChart ChartBase class, see the Help contents under LotusScript, LotusChart LotusScript Reference, By Category, Classes.

Base classes

ChartBase

Contained by

Class	Property
DrawObject	Chart

Freelance Graphics: Color class members

Properties

- Blue
- Green
- Red

Methods

- GetNearestIndex
- GetRGB
- SameColor

Functions

None

Events

None

Freelance Graphics: Color class

A color. You can specify a color using either the Item or RGBToColor methods of the Colors class.

Base classes

BaseObject

Contained by

Class	Property
Background	Color, BackColor
Border	Color
BulletProperties	Color
LineStyle	Color
Font	FontColor
TextProperties	ShadowColor

Freelance Graphics: Colors class members

Properties

Count

Methods

ColorToRGB
GetIndex
GetNearestColor
IsEmpty
Item
RGBtoColor

Functions

None

Events

None

Freelance Graphics: Colors class

The color library.

Base classes

BaseObject

Contained by

Class	Property
Application	Colors

Freelance Graphics: Document class members

Properties

Active
ActivePage
Application AS Application class
Author
Changed
Description
DocName
DocWindow AS DocWindow class
Embedded
FullName
IsOpen
Location
Pages AS Pages class
PageSelection AS PageSelection class
PageTransitionDelay
PageTransitionEffect
Parent
Path
ReadOnly
Selection AS Selection class
SmartLook
TemplatePageCount
ViewMode

Methods

Activate
AddToPageSelection
AddToSelection
Close
CopySelection
CreatePage
CreatePageFromTemplate
CutSelection
DeletePage
DeleteReviewer
Deselect
DistributeForComment
Export
FindObject
GotoPage
Import
OpenPresForCopy
Paste
PastePage
PasteSelectedPages
PublishToWeb
Print
PrintOut
RemoveFromSelection
RunDialog
Save
SaveAs
SaveAsToInternet
SaveAsToNotes
Select
SelectPageForCopy

SetViewMode
Show
StartGuidedTemplate
StopGuidedTemplate

Functions

None

Events

Activated
Created
Opened
PageCreated
PreClose
SaveAs
SavedAs
Saved
Save
SMCStarted

Freelance Graphics: Document class
Models a Freelance Graphics presentation. You can create an object of the Document class using the NewDocument method of the Application class.

Base classes
BaseObject

Contained by

Class	Property
Page	Document
Application	ActiveDocument

Usage
You can use the CurrentDocument predefined variable to reference the properties and methods of the currently selected Document object.

Freelance Graphics: Documents class members

Properties

Count

Methods

GetIndex
IsEmpty
Item

Functions

None

Events

None

Freelance Graphics: Documents class

A collection of Documents.

Base classes

BaseObject

Contained by

Class	Property
Application	Documents

Freelance Graphics: DocWindow class

A document window.

Base classes

BaseObject

Contained by

Class	Property
Document	DocWindow

Usage

You can use the CurrentDocWindow predefined variable to reference the properties and methods of the currently selected DocWindow object.

Freelance Graphics: DocWindow class members

Properties

Height
Width

Methods

Cascade
Close
Maximize
Minimize
Restore
Tile

Functions

None

Events

None

Freelance Graphics: DrawObject class

Any selectable object on a page in a presentation.

Base classes

BaseObject

Contained by

Class	Property
Page	Title

Usage

You can use the Selection or CurrentSelection predefined variable to reference the properties and methods of the currently selected DrawObject object.

Freelance Graphics: DrawObject class members

Properties

- Application
- Background AS Background class
- Border AS Border class
- BuildBullets
- Chart AS Chart class
- Delay
- Description
- DimPrevious
- ExeName
- Height
- IsChart
- IsGroup
- IsImage
- IsMedia
- IsOleObj
- IsPlacementBlock
- IsSelectable
- IsTable
- IsText
- IsValid
- Left
- LineStyle AS LineStyle class
- Media AS Media class
- Name
- ObjectType
- OleObj
- Page AS Page class
- Parent
- PlacementBlock AS PlacementBlock class
- PlayPriority
- Priority
- Table AS Table class
- TextBlock AS TextBlock class
- Top
- TransitionEffect
- VersionID
- WaitForClick
- Width

Methods

- AddPoint
- ConvertTo
- Copy
- Cut
- Flip
- GetObjectData
- Item
- Move
- PutIntoPlacementBlock
- Remove
- Replicate
- Rotate
- SetObjectData
- SetSSAction
- Stretch
- Ungroup

Functions

None

Events

None

Freelance Graphics: Font class members

Properties

Bold
Case
DoubleUnderline
FontColor
FontName
Italic
Overstrike
Size
SmallCaps
StrikeThrough
SubScript
SuperScript
Underline
WordDoubleUnderline
WordUnderline

Methods

RevertToStyle

Functions

None

Events

None

Freelance Graphics: Font class

The style of a selection or block of text.

Base classes

BaseObject

Contained by

Class	Property
TextBlock	Font
TextProperties	Font

Freelance Graphics: LineStyle class members

Properties

Color AS Color class
Pattern
Width

Methods

RevertToStyle

Functions

None

Events

None

Freelance Graphics: LineStyle class

The style of a line, arrow, or connector.

Base classes

BaseObject

Contained by

Class	Property
DrawObject	LineStyle

Freelance Graphics: Media class members

Properties

None

Methods

Play

StopPlay

Functions

None

Events

None

Freelance Graphics: Media class

Derived from DrawObject - a movie or animation. To create a Media object, use the CreateMovie method of the Page class.

Base classes

BaseObject

Contained by

Class	Property
DrawObject	Media

Freelance Graphics: Objects class members

Properties

Count

Methods

GetIndex
IsEmpty
Item

Functions

None

Events

None

Freelance Graphics: Objects class

A collection of DrawObjects.

Base classes

BaseObject

Contained by

Class	Property
Page	Objects

Freelance Graphics: OLEObject class
Controls an OLE object.

Base classes
DrawObject.

Contained by

Class	Property
DrawObject	Varient

Usage
Use this class to manipulate the OLE object, to find out its name, to find out its type, to convert it, and to put it into a placement block.

Freelance Graphics: OLEObject class members

Properties

Application
Background AS Background class
Border AS Border class
BuildBullets
Chart AS Chart class
Delay
Description
DimPrevious
ExeName
Height
IsChart
IsDraggable
IsGroup
IsImage
IsMedia
IsOleObj
IsPlacementBlock
IsSelectable
IsTable
IsText
IsValid
Left
LineStyle AS LineStyle class
Media AS Media class
Name
Object
OleObj
Page AS Page class
Parent
PlacementBlock AS PlacementBlock class
PlayPriority
Priority
Table AS Table class
TextBlock AS TextBlock class
Top
TransitionEffect
UserClassNameApplication
UserClassNameFull
UserClassNameShort
VersionID
WaitForClick
Width

Methods

Activate
AddPoint
ConvertTo
Copy
Cut
DoVerb
Flip
GetObjectData
Item
Move
PutIntoPlacementBlock
Remove

Replicate
Rotate
SetObjectData
Stretch
Ungroup

Functions

None

Events

None

Freelance Graphics: Page class

One page or slide in a presentation. To create a Page object, you can use either the CreatePage method or the CreatePageFromTemplate method of the Document class.

Base classes

BaseObject

Contained by

Class	Property
DrawObject	Page
Document	ActivePage

Usage

You can use the CurrentPage predefined variable to reference the properties and methods of the currently selected Page object.

Freelance Graphics: Page class members

Properties

AutoTime
Delay
Document AS Document class
Exclude
Layout
Name
Number
Objects AS Objects class
Selection AS Selection class
Sound
SpeakerNoteText
Title
TransitionEffect

Methods

CopyPage
CreateArrow
CreateChart
CreateComment
CreateLine
CreateMovie
CreateObject
CreateOval
CreateRect
CreateSymbol
CreateTable
CreateText
CutPage
DeleteSpeakerNote
FindNextObject
FindObject
GetSpeakerNoteMarkup
Move
Paste
PasteSpecial
Remove
Replicate

Functions

None

Events

Activated
Created

Freelance Graphics: Pages class members

Properties

Count

Methods

GetIndex

IsEmpty

Item

Functions

None

Events

None

Freelance Graphics: Pages class

A collection of Pages.

Base classes

BaseObject

Contained by

Class	Property
Document	Pages

Freelance Graphics: PageSelection class
The currently selected pages; derives methods and properties from Page.

Base classes
Page

Contained by

Class	Property
Application	PageSelection
Document	PageSelection

Freelance Graphics: PageSelection class members

Properties

SelectionCount

Methods

AddToSelection
ClearSelection
GetSelection
RemoveFromSelection
Select

Functions

None

Events

None

Freelance Graphics: PlacementBlock class members

Properties

- ID
- PromptText
- Type

Methods

- Activate
- BrowseDiagrams
- BrowseSymbols
- Insert

Functions

None

Events

- Clicked

Freelance Graphics: PlacementBlock class

Derived from DrawObject, otherwise known as a "Click here..." block.

Base classes

BaseObject

Contained by

Class	Property
DrawObject	PlacementBlock

Freelance Graphics: Preferences class

Freelance Graphics preferences.

Base classes

BaseObject

Contained by

Class	Property
Application	Preferences

Freelance Graphics: Preferences class members

Properties

- AutoSave
- AutoSaveInterval
- BackupDir
- BlackWhitePal
- BorderDisplay
- DisplayCoords
- DisplayDrawRuler
- DisplayGrid
- DisplayTextRuler
- OffsetReplicate
- ScanSpeed
- SkipWelcome
- SnapToGrid
- StartupView
- TemplateDir
- UndoEnabled
- WorkDir

Methods

None

Functions

None

Events

None

Freelance Graphics: Selection class members

Properties

- SelectedObjects
- SelectionCount

Methods

- AddToSelection
- Align
- ClearSelection
- Connect
- GetSelection
- Group
- RemoveFromSelection
- Select

Functions

None

Events

None

Freelance Graphics: Selection class

The currently selected draw objects; derives methods and properties from DrawObject.

Base classes

BaseObject

Contained by

Class	Property
Application	Selection
Page	Selection

Freelance Graphics: Table class

Derived from DrawObject - a Freelance Graphics table. To create a Table object, use the CreateTable method of the Page class.

Base classes

BaseObject

Contained by

Class	Property
DrawObject	Table

Freelance Graphics: Table class members

Properties

- ColCount
- RowCount

Methods

- DeleteCol
- DeleteRow
- GetCell
- InsertCol
- InsertRow

Functions

None

Events

None

Freelance Graphics: TextBlock class

Derived from DrawObject - a text block or text shape.

Base classes

DrawObject

Contained by

Class	Property
DrawObject	TextBlock

Freelance Graphics: TextBlock class members

Properties

BulletProperties AS BulletProperties class
Font AS Font class
Text
TextProperties AS TextProperties class

Methods

ApplyStyle
CreateStyle
EnterEditMode
GetBulletCount
GetMarkup
GetNthBullet
LeaveEditMode
RevertToStyle

Functions

None

Events

None

Freelance Graphics: TextProperties class

Level-specific text properties.

Base classes

BaseObject

Contained by

Class	Property
TextBlock	TextProperties

Freelance Graphics: TextProperties class members

Properties

FirstIndent
Font AS Font class
HorizontalAlignment
LineLead
ParaIndent
Paralead
RightIndent
ShadowColor AS Color class
ShadowDepth
ShadowDirection
VerticalAlignment

Methods

None

Functions

None

Events

None

RGB value (defined)

A type Long number specifying the amount of red, green, and blue tint in a 24-bit color. The high-order byte is 0 and is unused. The three low-order bytes each contain a binary value from 0 (no tint) to 255 (maximum tint), with the lowest-order byte representing blue, the next byte green, and the next byte red.

twips (defined)

A twip is a screen-independent unit of measurement (unlike a pixel, which is screen-dependent). There are 1440 twips to an inch and 567 twips to a centimeter. In LotusScript, all Freelance Graphics coordinates and sizes are specified in twips.

Freelance Graphics: Activated event (Document)

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_ACTIVATED_DOC_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics has finished activating a presentation.

Internal syntax

Activated(Source As Document)

Parameters

Source As Document

Represents the active presentation as a Document object.

Usage

Use this event to run a script when Freelance Graphics has completed activating a presentation file (.PRZ).

```
' Example: Activated event (Document)
'// In this example, a message box comes up when the user activates the document
or  //
'// opens the document. //
Sub Activated(Source As Document)
     MessageBox "The document has just been opened or activated."
End Sub
```

Freelance Graphics: Activated event (Page)

{button ,AL(`;H_FLG_PAGE_CLASS',0)} See list of classes

{button ,AL(`H_FLG_ACTIVATED_PAGE_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics has finished activating a page.

Internal syntax

Activated(Source As Page)

Parameters

Source As Page

Represents the activated page as a Page object.

Usage

Use this event to run a script when Freelance Graphics has completed activating a page. That page is now the current page.

```
' Example: Activated event (Page)
'// In this example, a message box comes up when the user activates the page.//
Sub Activated(Source As Page)
     MessageBox "The page has just been activated."
End Sub
```

```
' Example: Clicked event
'// In this example, a message box comes up when the placement block that this //
'// script is attached to has been clicked. //
Sub Clicked(Source As PlacementBlock)
     MessageBox " The 'Click here...' block has just been clicked. "
End Sub
```

Freelance Graphics: Clicked event

{button ,AL(`;H_FLG_PLACEMENTBLOCK_CLASS',0)} See list of classes

{button ,AL(`H_FLG_CLICKED_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics detects that a given placement block has been clicked.

Internal syntax

Clicked(Source As PlacementBlock)

Parameters

Source As PlacementBlock

Represents the clicked placement block as a PlacementBlock object.

Usage

Use this event to run a script when the placement block has been clicked.

Freelance Graphics: Created event (Document)

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_CREATED_DOCUMENT_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics has finished creating a new presentation document (.PRZ file).

Internal syntax

Created(NewDocument As Document)

Parameters

NewDocument As Document

Represents the newly created presentation as a Document object.

Usage

Use this event to run a script when Freelance Graphics has created a document (presentation file). Note that although Freelance Graphics has created the document, it has not yet been named or saved.

Because a presentation (.PRZ) file inherits scripts from SmartMaster content files (.SMC), the most effective use of the Created event is when it is employed in a SmartMaster content (.SMC) file. It is meaningless in a .PRZ file.

```
' Example: Created event (Document)
'// In this example, a message box comes up when a new document has been created.//
Sub Created(Source As Document)
     MessageBox " A new document has been created. "
End Sub
```

Freelance Graphics: Created event (Page)

{button ,AL(`;H_FLG_PAGE_CLASS',0)} See list of classes

{button ,AL(`H_FLG_CREATED_PAGE_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics has finished creating a page.

Internal syntax

Created(Source As Page)

Parameters

Source As Page

Represents the newly created page as a Page object.

Usage

Use this event to run a script when Freelance Graphics has created a page.

Because a presentation (.PRZ) file inherits scripts from SmartMaster content files (.SMC), the most effective use of the Created event is when it is employed in a SmartMaster content (.SMC) file, it is meaningless in a .PRZ file.

```
' Example: Created event (Page)
'// In this example, a message box comes up when a new page has been created.//
Sub Created(Source As Page)
     MessageBox " A new page has been created. "
End Sub
```

```
' Example: SMCStarted event
'// In this example, a message box comes up when a SmartMaster with content is //
'// chosen as a basis for creating a presentation page. It can be used to explain what
a //
'// SmartMaster with content can be used for.//
Sub SMCStarted(Source As Document)
     MessageBox " You are about to use a SmartMaster with content as the basis" +_
                  "for your presentation page. "
End Sub
```

Freelance Graphics: SMCStarted event

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_GTSTARTED_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics uses a SmartMaster content topic as the basis for a presentation page.

Internal syntax

SMCStarted(Source As Document)

Parameters

Source As Document

Represents the active document as a Document object.

Usage

Use this event to run a script when Freelance Graphics receives a command to use a content topic for a presentation. This event is triggered when the user chooses File - New and selects a SmartMaster with content, or when the user switches over from not using a SmartMaster content topic to using a SmartMaster content topic by choosing Presentation - SmartMaster Content - Select a Topic.

```
' Example: Opened event
'// In this example, a message box comes up when Freelance Graphics has just opened //
'//  a presentation (.prz) file.//
Sub Opened(Source As Document)
     MessageBox " The presentation is now open. "
End Sub
```

Freelance Graphics: Opened event

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_OPENED_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics has finished opening an existing presentation file.

Internal syntax

Open(Source As Document)

Parameters

Source As Document

Represents the just opened presentation file as a Document object.

Usage

Use this event to run a script when Freelance Graphics has opened a presentation.

```
' Example: PageCreated event
'// In this example, a message box comes up when Freelance Graphics has created a new
page.//
Sub PageCreated(Source As Page)
     MessageBox " A new page has just been created. "
End Sub
```

Freelance Graphics: PageCreated event

{button ,AL(`;H_FLG_PAGE_CLASS',0)} See list of classes

{button ,AL(`H_FLG_PAGECREATED_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics has finished creating a page.

Internal syntax

PageCreated(Source As Page)

Parameters

Source As page

Represents the newly created page as a Page object.

Usage

Use this event to run a script when Freelance Graphics has created a page.

```
' Example: PreClose event
'// In this example, a message box comes up when Freelance Graphics receives an //
'// instruction to close the document. //
Sub PreClose(Source As Document)
     MessageBox " The document is about to close. "
End Sub
```

Freelance Graphics: PreClose event

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_CLOSE_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics receives an instruction to close the document (.PRZ file).

Internal syntax

PreClose(Source As Document)

Parameters

Source As Document

Represents the document to be closed as a Document object.

Usage

Use this event to run a script when Freelance Graphics receives a command to close a document.

Freelance Graphics: SaveAs event

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SAVEAS_EVENT_EXSCRIPT',1)} See example

Raised just before Freelance Graphics writes to another file type (such as a .BMP file) or as a presentation (.PRZ) file.

Internal syntax

SaveAs(Source As Document)

Parameters

Source As Document

Represents the presentation to be saved as a Document object.

Usage

Use this event to run a script when the user chooses File - Save As and saves the presentation as a .PRZ file or as as another file type (such as a .BMP file). The event is raised just before Freelance Graphics actually writes the file.

```
' Example: SaveAs event
'// In this example, a message box comes up when Freelance Graphics receives a
command//
'// to save a presentation as a .PRZ file or as another file type.//
Sub SaveAs(Source As Document)
     MessageBox " Freelance Graphics is about to save the presentation " +_
                "as a file type of your choice. "
End Sub
```

```
' Example: SavedAs event
'// In this example, a message box comes up when Freelance Graphics has saved //
'// a presentation as a .PRZ file or as another file type.//
Sub SavedAs(Source As Document)
     MessageBox " The presentation has been saved as the file type you chose. "
End Sub
```

Freelance Graphics: SavedAs event

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SAVEDAS_EVENT_EXSCRIPT',1)} See example

Raised each time Freeelance Graphics has finished saving a presentation as a presentation (.PRZ) file or as another file type.

Internal syntax

SavedAs(Source As Document)

Parameters

Source As Document

Represents the presentation just saved as a Document object.

Usage

Use this event to run a script when the user chooses File - Save As and saves the presentation as a .PRZ file or as another file type (such as a .BMP file). The event is raised when Freelance Graphics has finished writing to a file.

Freelance Graphics: Saved event

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SAVED_EVENT_EXSCRIPT',1)} See example

Raised when Freelance Graphics has finished saving a presentation (.PRZ) file.

Internal syntax

Saved(Source As Document)

Parameters

Source As Document

Represents the presentation that has just been saved as a Document object.

Usage

Use this event to run a script when the user chooses File - Save. The event is raised after Freelance Graphics has finished writing the file.

```
' Example: Saved event
'// In this example, a message box comes up when Freelance Graphics has saved //
'// a presentation.//
Sub Saved(Source As Document)
     MessageBox " The presentation has been saved. "
End Sub
```

Freelance Graphics: Save event

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SAVE_EVENT_EXSCRIPT',1)} See example

Raised each time Freelance Graphics receives an instruction to save a presentation and just before Freelance Graphics actually writes to the file.

Internal syntax

Save(Source As Document)

Parameters

Source As Document

Represents the presentation about to be saved as a Document object.

Usage

Use this event to run a script when the user chooses File - Save. The event is raised just before Freelance Graphics writes to the file.

```
' Example: Save event
'// In this example, a message box comes up when Freelance Graphics receives a
command//
'// to save a presentation.//
Sub Save(Source As Document)
     MessageBox " Freelance Graphics is about to save this file. "
End Sub
```

Freelance Graphics LotusScript Events A-Z

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

```
' Example: Activate method
' The following examples show three ways of using the activate method.
' This example requires that a least two presentations be open.
' It makes the second presentation the active document.
CurrentApplication.Documents(2).Activate
' or, next example:
' This example requires that a text placement block be on the page.
[TextPlacementBlock1].Activate
' or, next example:
' This example puts a Paintbrush object on the page.
MyOLE As Variant
Set MyOLE = CurrentPage.CreateObject("Paintbrush Picture")
MyOLE.Activate
```

Freelance Graphics: Activate method

{button ,AL(`;H_FLG_DOCUMENT_CLASS;H_FLG_OLEOBJECT_CLASS;H_FLG_PLACEMENTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ACTIVATE_METHOD_EXSCRIPT',1)} See example

Simulate a click action on a document, a "Click here" block,   or an OLE object. The object becomes the current object and is brought to the foreground.

Syntax

documentobject.**Activate**

or

placementblockobject.**Activate**

or

oleobject.**Activate**

Parameters

None

Return values

None

Usage

When used on an instance of the Document class, the Activate method brings the document to the front. When used on an instance of the PlacementBlock class, the Activate method simulates a click on a "Click here" block. When used on an instance of the OLEObject class, the Activate method launches the server associated with the embedded object.

```
' Example: AddPoint method
Dim LineObject As DrawObject
'Creates a line object with the given coordinates in twips
Set LineObject = CurrentPage.CreateLine (400, 1000, 1000, 2000)
'Adds an additional point to the line object where x and y are the coordinates in
twips
LineObject.AddPoint 1, 6000, 1000
```

Freelance Graphics: AddPoint method

{button ,AL(`;H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ADDPOINT_METHOD_EXSCRIPT',1)} See example

Add a point to the object.

Syntax

drawobject.**AddPoint(***segment*, *x*, *y***)**

Parameters

segment as Integer

Identifies the point after which you want to add the new point. (The first drawn point is 1.) If the object in question is not a polygon, line, arrow, or curve, this value is ignored.

x as Integer

Horizontal coordinate of the point relative to the left edge of the window, in twips.

y as Integer

Vertical coordinate of the point relative to the bottom of the window, in twips.

Return values

None

```
' Example: AddToPageSelection method
' Create a new page, then add it to the selected pages.
Dim MyPage As Page
Set MyPage = CurrentDocument.CreatePage("My Title Page",1)
CurrentDocument.AddToPageSelection MyPage
```

Freelance Graphics: AddToPageSelection method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ADDTOPAGESELECTION_METHOD_EXSCRIPT',1)} See example

Add a page to the set of selected pages in the document.

Syntax

documentobject.**AddToPageSelection(***pageobject***)**

Parameters

pageobject as Page

Any page object.

Return values

None

Freelance Graphics: AddToSelection method
{button ,AL(`H_FLG_DOCUMENT_CLASS;H_FLG_PAGESELECTION_CLASS;H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ADDTOSELECTION_METHOD_EXSCRIPT',1)} See example

Add the specified object to the set of selected documents, pages, or objects.

Syntax
object.**AddToSelection(***drawobject***)**

Parameters
drawobject as DrawObject

Any DrawObject object.

Return values
None

```
' Example: AddToSelection method
' There are two examples here.  In both examples, you must
' have a rectangle, MyRect, on the current page and
' you must have that rectangle selected.
CurrentPage.Selection.AddToSelection MyRect
' or, next example:
Selection.AddToSelection MyRect
```

Freelance Graphics: Align method

{button ,AL(`H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ALIGN_METHOD_EXSCRIPT',1)} See example

Align the currently selected objects.

Syntax

selectionobject.**Align(***aligntype*, *centeronpage***)**

Parameters

aligntype as Variant (Enumerated)

Value	Description
$AlignLeft	Align left sides
$AlignRight	Align right sides
$AlignTop	Align tops
$AlignBottom	Align bottoms
$AlignRow	Center in a row
$AlignColumn	Center in a column
$AlignPoint	Center on a point

centeronpage as Integer (Boolean)

Value	Description
TRUE (-1)	Center on page
FALSE (0)	Do not center on page

Return values

None

```
' Example: Align method
Selection.Align $AlignLeft, False
```

```
' Example: ApplyStyle method
Dim TextObject As DrawObject
' Create a text block object
Set TextObject = CurrentPage.CreateText(4000, 4000, 3000, 3000)
' Add text to the text block
TextObject.TextBlock.Text = "This is the title"
' Apply the Presentation Title Named Style to the text block
TextObject.TextBlock.ApplyStyle("Presentation Title")
```

Freelance Graphics: ApplyStyle method

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_APPLYSTYLE_METHOD_EXSCRIPT',1)} See example

Apply the specified style name to the text block.

Syntax

textblockobject.**ApplyStyle(***stylename***)**

Parameters

stylename as String

Name to apply to the style.

Return values

None

Freelance Graphics: BrowseDiagrams method

{button ,AL(`H_FLG_PLACEMENTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BROWSEDIAGRAMS_METHOD_EXSCRIPT',1)} See example

Launch the Diagram browser using the specified diagram file. When the user clicks OK in the browser, inserts the currently selected diagram in the "Click here to add diagram" block.

Note This method only adds diagrams and works only with diagram placement blocks. To add clip art, see the BrowseSymbols method.

Syntax

placementblockobject.**BrowseDiagrams(***filename***)**

Parameters

filename as String

Optional name of the diagram file. If omitted, defaults to the first .DGM file in the \SMASTERS\FLG directory.

Return values

None

```
' Example: BrowseDiagrams method
' In this example you must have a diagram placement block on the page.
Forall obj In CurrentPage.objects
   If Obj.IsPlacementBlock Then
      Obj.PlacementBlock.BrowseDiagrams("Branch.dgm")
   End If
End Forall
```

Freelance Graphics: BrowseSymbols method

{button ,AL(`H_FLG_PLACEMENTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BROWSESYMBOLS_METHOD_EXSCRIPT',1)} See example

Launch the Clip Art browser using the specified symbol file. When the user clicks OK in the browser, insert the currently selected symbol in the "Click here to add clip art" block.

Note This method only creates clip art and words only with clip art placement blocks. To create diagrams, see the BrowseDiagrams method.

Syntax

placementblockobject.**BrowseSymbols(***filename***)**

Parameters

filename as String

Optional name of the symbols file. If omitted, defaults to the first .SYM file in the \SMASTERS\FLG directory.

Return values

None

```
' Example: BrowseSymbols method
' In this example you must have a clip art placement block on the page.
Forall obj In CurrentPage.objects
   If Obj.IsPlacementBlock Then
      Obj.PlacementBlock.BrowseSymbols("people.sym")
   End If
End Forall
```

```
' Example: Cascade method
CurrentApplication.Cascade
```

Freelance Graphics: Cascade method

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CASCADE_METHOD_EXSCRIPT',1)} See example

Cascades all document windows within the Freelance Graphics application window.

Syntax

applicationwindowobject.**Cascade**

Parameters

None

Return values

None

Freelance Graphics: ClearSelection method
{button ,AL(`H_FLG_PAGESELECTION_CLASS;H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CLEARSELECTION_METHOD_EXSCRIPT',1)} See example

Deselect all selected items.

Syntax
object.**ClearSelection**

Parameters
None

Return values
None

```
' Example: ClearSelection method
Selection.ClearSelection
```

```
' Example: CloseWindow method
CurrentApplication.Save
CurrentApplication.CloseWindow
```

Freelance Graphics: CloseWindow method

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CLOSEWINDOW_METHOD_EXSCRIPT',1)} See example

Closes the Freelance Graphics application.

Syntax

applicationobject.**CloseWindow(***closewindow***)**

Parameters

closewindow as Integer

Optional and ignored.

Return values

None

Status codes

None

Freelance Graphics: Close method

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCUMENT_CLASS;H_FLG_DOCWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CLOSE_METHOD_EXSCRIPT',1)} See example

Close the Freelance Graphics application window (exit Freelance Graphics), or close a document or document window.

Syntax

*documentobject***.Close***(savechanges, docname, location)*

or

*appwindowbject***.Close**

or

*docwindowobject***.Close(***exitapplication***)**

Parameters

savechanges as Integer

Optional. If True (non-zero), changes are always saved. If False (zero), changes are never saved. If omitted, the user is consulted if the file has been changed.

docname as String

Optional. Document name.

location as String

Optional. Path.

exitapplication as *Integer*

Optional. If False (zero),   saves the document (if this is the first time the file is saved, opens the Save As dialog box, so user can enter a file name). If True (non-zero) or omitted, file closes without saving the document.

Return values

None

Usage

The Close method works slightly differently for each of the classes where it is available.

In the Document.Close case: If the first parameter, *savechanges*, is omitted, the user is consulted if the file is changed. If *savechanges* is True, the file is saved. If *savechanges* is False, the file is closed without being saved.

In the ApplicationWindow.Close case: The application simply closes, parameters are ignored. Nothing is saved.

In the DocWindow.Close case: There is only one parameter in this case. If the parameter, *exitapplication*, is False (zero), the file is saved. If the parameter, *exitapplication*, is True or omitted, the file is closed without being saved.

```
' Example: Close method
' An example for each of the classes that the Close method pertains to.
CurrentDocument.Close
' or, next example:
CurrentDocWindow.Close
' or, next example:
CurrentApplicationWindow.Close
```

Freelance Graphics: ColorToRGB method

{button ,AL(`H_FLG_COLORS_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COLORTORGB_METHOD_EXSCRIPT',1)} See example

Return the RGB value of the color produced by the current values of the Red, Green, and Blue properties of a Color object.

Syntax

colorsobject.**ColorToRGB(***colorobject***)**

Parameters

colorobject as Color

The Color object whose RGB value you want returned.

Return values

Long, RGB value.

```
' Example: ColorToRGB method
' In this example you must have previously identified a color as ThatColor.
Dim MyColorRGB As Long
MyColorRGB = CurrentApplication.Colors.ColorToRGB(ThatColor)
```

```
' Example: Connect method
Selection.Connect
```

Freelance Graphics: Connect method
{button ,AL(`H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CONNECT_METHOD_EXSCRIPT',1)} See example

Connect the currently selected line objects.

Syntax
*selection***.Connect**

Parameters
None

Return values
None

Freelance Graphics: ConvertTo method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CONVERTTO_METHOD_EXSCRIPT',1)} See example

Convert the object to lines or polygons.

Syntax

drawobject.**ConvertTo(***conversiontype***)**

Parameters

conversiontype as Integer (Enumerated)

Value	Description
$ConvertToPolygons	Create one or more polygons from the object
$ConvertToLines	Create one or more lines from the object

Return values

None

```
' Example: ConvertTo method
Dim MyRect As DrawObject
Set MyRect = CurrentPage.CreateRect
MyRect.ConvertTo($ConvertToLines)
```

```
' Example: CopyPage method
CurrentPage.CopyPage
```

Freelance Graphics: CopyPage method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COPYPAGE_METHOD_EXSCRIPT',1)} See example

Copy a page to the Clipboard.

Syntax

pageobject.**CopyPage**

Parameters

None

Return values

None

```
' Example: CopySelection method
CurrentDocument.CopySelection
```

Freelance Graphics: CopySelection method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COPYSELECTION_METHOD_EXSCRIPT',1)} See example

Copy the selection to the Clipboard.

Syntax

documentobject.**CopySelection**

Parameters

None

Return values

None

```
' Example: Copy method
' Two examples, the first requires that something on the page
' be selected.
Selection.Copy
' or, next example:
Dim MyRect As DrawObject
' Create a rectangle, then copy it.
Set MyRect = CurrentPage.CreateRect
MyRect.Copy
```

Freelance Graphics: Copy method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COPY_METHOD_EXSCRIPT',1)} See example

Copy an object to the Clipboard.

Syntax

drawobject.**Copy**

Parameters

None

Return values

None

Freelance Graphics: CreateArrow method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATEARROW_METHOD_EXSCRIPT',1)} See example

Draw an arrow on the page.

Syntax

pageobject.**CreateArrow(***xstart***,** *ystart***,** *xfinish***,** *yfinish***)**

Parameters

xstart as Integer

(Optional, see note) Starting horizontal coordinate of the arrow, in twips.

ystart as Integer

(Optional, see note) Starting vertical coordinate of the arrow, in twips.

xfinish as Integer

(Optional, see note) Ending horizontal coordinate of the arrow, in twips.

yfinish as Integer

(Optional, see note) Ending vertical coordinate of the arrow, in twips.

Note If you omit the parameters, the arrow will be centered on the page.

Return values

An instance of the DrawObject class (the drawn arrow).

```
' Example: CreateArrow method
Dim MyArrow As DrawObject
Set MyArrow = CurrentPage.CreateArrow(1000, 1000, 3000, 3000)
```

```
' Example: CreateChart method
Dim MyChart As Chart
Set MyChart = CurrentPage.CreateChart(4000,10000,6000,6000)
```

Freelance Graphics: CreateChart method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATECHART_METHOD_EXSCRIPT',1)} See example

Create a chart on the page.

Note For more information on working with charts in LotusScript, search for ChartBase in the Help Index.

Syntax

*pageobject***.CreateChart(***x***,** *y***,** *width***,** *height***)**

Parameters

x as Integer

(Optional, see note) Right edge of the chart object relative to the left edge of the window, in twips.

y as Integer

(Optional, see note) Bottom edge of the chart object relative to the bottom of the window, in twips.

width as Integer

(Optional, see note) Width in twips.

height as Integer

(Optional, see note) Height in twips.

Note If you omit the parameters, the chart will be centered on the page.

Return values

An instance of the DrawObject class (the drawn chart).

```
' Example: CreateComment method
CurrentPage.CreateComment "Four score and seven years ago"
```

Freelance Graphics: CreateComment method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATECOMMENT_METHOD_EXSCRIPT',1)} See example

Create TeamReview comments on a page.

Syntax

pageobject.**CreateComment(***x, y, width, height***)**

Parameters

x as Integer

(Optional, see note) Right edge of the comment object relative to the left edge of the window, in twips.

y as Integer

(Optional, see note) Bottom edge of the comment object relative to the bottom of the window, in twips.

width as Integer

(Optional, see note) Width in twips.

height as Integer

(Optional, see note) Height in twips.

Note If you omit the parameters, the comment will be centered on the page.

Return values

An instance of the DrawObject class (the TeamReview comments).

```
' Example: CreateLine method
Dim MyLine As DrawObject
Set MyLine = CurrentPage.CreateLine(1000, 1000, 3000, 3000)
```

Freelance Graphics: CreateLine method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATELINE_METHOD_EXSCRIPT',1)} See example

Draw a line on the page.

Syntax

pageobject.**CreateLine(***xstart***,** *ystart***,** *xfinish***,** *yfinish***)**

Parameters

xstart as Integer

(Optional, see note) Starting horizontal coordinate of the line, in twips.

ystart as Integer

(Optional, see note) Starting vertical coordinate of the line, in twips.

xfinish as Integer

(Optional, see note) Ending horizontal coordinate of the line, in twips.

yfinish as Integer

(Optional, see note) Ending vertical coordinate of the line, in twips.

Note If you omit the parameters, the line will be centered on the page.

Return values

An instance of the DrawObject class (the drawn line).

Freelance Graphics: CreateMovie method
{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes
{button ,AL(`H_FLG_CREATEMOVIE_METHOD_EXSCRIPT',1)} See example
Add a movie to the page.

Syntax
pageobject.**CreateMovie(***x, y, width, height, filename***)**

Parameters
x as Integer

(Optional, see note) Left edge of the movie window relative to the left edge of the window, in twips.

y as Integer

(Optional, see note) Bottom edge of the movie window relative to the bottom of the window, in twips.

width as Integer

(Optional, see note) Width in twips.

height as Integer

(Optional, see note) Height in twips.

filename as String

Name of the file containing the movie to be added to the page.

Note If you omit the optional parameters, the movie will be located in a default position defined by the product.

Return values
An instance of the DrawObject class (the movie object).

```
' Example: CreateMovie method
Dim MovieObject  As Drawobject
Set MovieObject = CurrentPage.CreateMovie("dinosaur.aim")
```

Freelance Graphics: CreateObject method

{button ,AL(`;H_FLG_PAGE_CLASS',0)} See list of classes

{button ,AL(`H_FLG_CREATEOBJECT_METHOD_EXSCRIPT',1)} See example

Controls the creation of an OLE object. The OLE object is an instance of the DrawObject class.

Syntax

PageObject**.CreateObject**(*class type|filename, link, displayicon, x, y, w, h)*

Parameters

classtype as String

Optional only if the second parameter is present, otherwise this parameter is manditory. Input parameter. The type of object that you want to embed in a presentation. The list of object types includes a Bitmap Image, Paintbrush Picture, and so on. In Freelance Graphics choose Create - Object to see the list of objects available to you. The list is dependent on those OLE applications that you have on your system. Use the program name as it appears in the Windows registry or you can use the progid from the registry.

filename as String

Optional only if the first parameter is present, otherwise this parameter is manditory. Input parameter. If creating the object from a file, use this parameter to show path and name of the file.

link as Integer

Optional. Input. True (link) or false (embed), where false is zero, and true is any non-zero integer.

displayasicon as Integer

Optional. Input. True (displayed as an icon) or false (displayed as a metafile representation). Where false is zero, and true is any positive integer.

x as Integer

(Optional, see note) Input. Horizontal coordinate measured in twips. The default value results in an object roughly centered on the page.

y as Integer

(Optional, see note) Input. Vertical coordinate measured in twips. The default value results in an object roughly centered on the page.

w as Integer

(Optional, see note) Input. Width of area taken up by the OLE object measured in twips. The default value results in an object roughly centered on the page.

h as Integer

(Optional, see note) Input. Height of the area taken up by the OLE object measured in twips. The default value results in an object roughly centered on the page.

Note If you omit the parameters, the object will be centered on the page.

Return values

Returns a OLEObject class reference.

Usage

Either the first or second parameter must be present.

```
' Example: CreateObject method
' There are two examples:
' In the first, you create an instance of
' an OLE object (a Paintbrush picture).
Dim MyOLE As Variant
Set MyOLE = CurrentPage.CreateObject ("PaintBrush Picture")
' In the second, you launch an instance
' of the Lotus Draw Component as an OCX object.
Dim MyOLE As Variant
Set MyOLE = CurrentPage.CreateObject ("Lotus Draw/Diagram Component")
```

```
' Example: CreateOval method
Dim MyOval As DrawObject
Set MyOval = CurrentPage.CreateOval(1000, 1000, 3000, 3000)
```

Freelance Graphics: CreateOval method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATEOVAL_METHOD_EXSCRIPT',1)} See example

Draw an oval on the page.

Syntax

pageobject.**CreateOval(***xcenter***,** *ycenter***,** *width***,** *height***)**

Parameters

xcenter as Integer

(Optional, see note) Horizontal coordinate starting at the top-left "corner" of the oval, in twips.

ycenter as Integer

(Optional, see note) Vertical coordinate starting at the top-left "corner" of the oval, in twips.

width as Integer

(Optional, see note) Width of the oval, in twips.

height as Integer

(Optional, see note) Height of the oval, in twips.

Note If you omit the parameters, the oval will be centered on the page.

Return values

An instance of the DrawObject class (the drawn oval).

```
' Example: CreatePageFromTemplate method
' Creates a new file using BusRev.Smc and
' uses CreatePageFromTemplate to add the
' third template page to the new document
CurrentApplication.NewDocument ,,,"BusRev.Smc"
Dim DocObject As Document
Set DocObject = CurrentDocument
DocObject.CreatePageFromTemplate "New Page", 3
```

Freelance Graphics: CreatePageFromTemplate method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATEPAGEFROMTEMPLATE_METHOD_EXSCRIPT',1)} See example

Create a new page using a content topic.

Syntax

*document***.CreatePageFromTemplate** *pagetitle***,** *templateindex*

Parameters

pagetitle as String

Name for the new page.

templateindex as Integer

Identifies the content topic page type (the first type is 1).

Return values

An instance of the Page class.

Freelance Graphics: CreatePage method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATEPAGE_METHOD_EXSCRIPT',1)} See example

Create a new page.

Syntax

documentobject.**CreatePage** *pagetitle*, *masterindex*

Parameters

pagetitle as String

Name for the new page

masterindex as Integer

Identifies the SmartLook to be used (1--refers to the first one in the list of SmartLooks that are on your system; 2--refers to the second, and so on). To see the list of SmartLooks: in the main menu, choose Presentation - Choose a Different SmartMaster Look.

Return values

An instance of the Page class.

```
' Example: CreatePage method
Dim Page1 As Page
Set Page1 = CurrentDocument.CreatePage("My Title Page",1)
```

```
' Example: CreateRect method
' Create a rectangle using specific dimensions.
Dim MyRect As DrawObject
Set MyRect = CurrentPage.CreateRect(1000, 1000, 2000, 2000)
```

Freelance Graphics: CreateRect method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATERECT_METHOD_EXSCRIPT',1)} See example

Draw a rectangle on the page.

Syntax

pageobject.**CreateRect(***xcenter***,** *ycenter***,** *width***,** *height***)**

Parameters

xcenter as Integer

(Optional, see note) Horizontal coordinate starting at the top-left corner of the rectangle, in twips.

ycenter as Integer

(Optional, see note) Vertical coordinate starting at the top-left corner of the rectangle, in twips.

width as Integer

(Optional, see note) Width of the rectangle, in twips.

height as Integer

(Optional, see note) Height of the rectangle, in twips.

Note If you omit the parameters, the rectangle will be centered on the page.

Return values

An instance of the DrawObject class (the drawn rectangle).

```
' Example: CreateStyle method
' This script creates a text block, fills it with text,
' and creates a named style for it
Dim TextObj As DrawObject
Set TextObj = CurrentPage.CreateText
Textobj.Text = "Text in this text block"
TextObj.TextBlock.CreateStyle("Mine")
```

Freelance Graphics: CreateStyle method
{button ,AL(`H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATESTYLE_METHOD_EXSCRIPT',1)} See example

Create a new named style based on the attributes of the text block.

Syntax
textblockobject.**CreateStyle(***stylename***)**

Parameters
stylename as String

Name of the new style.

Return values
None

Freelance Graphics: CreateSymbol method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATESYMBOL_METHOD_EXSCRIPT',1)} See example

Add a symbol to the page.

Syntax

pageobject.**CreateSymbol(***filename***,** *index, x, y, width, height***)**

Parameters

filename as String

Name of the file containing the symbol.

index as Integer

Sequence number of the symbol in the file.

x as Integer

(Optional, see note) Left edge of the symbol relative to the left edge of the window, in twips.

y as Integer

(Optional, see note) Bottom edge of the symbol relative to the bottom of the window, in twips.

width as Integer

(Optional, see note) Width in twips.

height as Integer

(Optional, see note) Height in twips.

Note If you omit the optional parameters, the symbol will be centered on the page.

Return values

An instance of the DrawObject class (the symbol object).

```
' Example: CreateSymbol method
' Adds the third symbol picture from animals.sym
' to the current page.
CurrentPage.CreateSymbol "Animals.sym", 3
```

```
' Example: CreateTable method
'Creates a 4 X 4 table using the first table type.
CurrentPage.CreateTable 1, 4, 4
```

Freelance Graphics: CreateTable method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATETABLE_METHOD_EXSCRIPT',1)} See example

Create a table on the page.

Syntax

pageobject.**CreateTable(***type***,** *rows***,** *columns, x, y, width, height***)**

Parameters

type as Integer

See the Create Table dialog's Table Gallery for examples of the following grid types.

Value	Description
1	Full grid
2	Grid except for first row and column
3	Outline around outside of table
4	No grid lines

rows as Integer

Number of rows.

columns as Integer

Number of columns.

x as Integer

(Optional, see note) Left edge of the table relative to the left edge of the window, in twips.

y as Integer

(Optional, see note) Bottom edge of the table relative to the bottom of the window, in twips.

width as Integer

(Optional, see note) Width in twips.

height as Integer

(Optional, see note) Height in twips.

Note If you omit the optional parameters, the table will be centered on the page.

Return values

An instance of the Table class.

Freelance Graphics: CreateText method
{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CREATETEXT_METHOD_EXSCRIPT',1)} See example

Create a text block.

Syntax
pageobject.**CreateText(***x*, *y*, *width*, *height***)**

Parameters
x as Integer

(Optional, see note) Left edge of the text block relative to the left edge of the window, in twips.

y as Integer

(Optional, see note) Top edge of the text block relative to the bottom of the window, in twips.

width as Integer

(Optional, see note) Width in twips.

height as Integer

(Optional, see note) Height in twips.

Note If you omit the parameters, the text will be centered on the page.

Return values
An instance of the DrawObject class.

```
' Example: CreateText method
' Creates a text block and fills it with text.
Dim TextObj As DrawObject
Set TextObj = CurrentPage.CreateText
TextObj.Text = "This text is inserted in the text block."
```

Freelance Graphics: CutPage method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CUTPAGE_METHOD_EXSCRIPT',1)} See example

Cut a Page object and place it on the Clipboard.

Syntax

pageobject.**CutPage**

Parameters

None

Return values

None

```
' Example: CutPage method
CurrentPage.CutPage
```

Freelance Graphics: CutSelection method
{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CUTSELECTION_METHOD_EXSCRIPT',1)} See example

Cut the current selection and place it on the Clipboard.

Syntax
documentobject.**CutSelection**

Parameters
None

Return values
None

```
' Example: CutSelection method
CurrentDocument.CutSelection
```

```
' Example: Cut method
' Create a rectangle, then cut it.
Dim MyRect As DrawObject
Set MyRect = CurrentPage.CreateRect()
MyRect.Cut
```

Freelance Graphics: Cut method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CUT_METHOD_EXSCRIPT',1)} See example

Cut a DrawObject and place it on the Clipboard.

Syntax

drawobject.**Cut**

Parameters

None

Return values

None

Freelance Graphics: DeleteCol method

{button ,AL(`H_FLG_TABLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DELETECOL_METHOD_EXSCRIPT',1)} See example

Delete a specified column from the table.

Syntax

tableobject.**DeleteCol(***column***)**

Parameters

column as Integer

Number of the column to delete.

Return values

None

```
' Example: DeleteCol method
' Creates a 4 X 4 table and deletes the second column.
Dim TableObj As DrawObject
Set TableObj = CurrentPage.CreateTable (1, 4, 4)
TableObj.DeleteCol 2
```

Freelance Graphics: DeletePage method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DELETEPAGE_METHOD_EXSCRIPT',1)} See example

Delete the specified page.

Syntax

documentobject.**DeletePage(***pageobject***)**

Parameters

pageobject as Page

The page to delete.

Return values

None

```
' Example: DeletePage method
' Deletes the current page in a presentation.
Dim PageObj As Page
Set PageObj = CurrentPage
CurrentDocument.DeletePage PageObj
```

```
' Example: DeleteReviewer method
' Opens the delete reviewer dialog for the current document.
Dim DocObj As Document
Set DocObj = CurrentDocument
DocObj.DeleteReviewer
```

Freelance Graphics: DeleteReviewer method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DELETEREVIEWER_METHOD_EXSCRIPT',1)} See example

Open the dialog box used to delete a TeamReview reviewer.

Syntax

documentobject.**DeleteReviewer**

Parameters

None

Return values

None

Freelance Graphics: DeleteRow method

{button ,AL(`H_FLG_TABLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DELETEROW_METHOD_EXSCRIPT',1)} See example

Delete a specified row from the table.

Syntax

tableobject.**DeleteRow(***row***)**

Parameters

row as Integer

The number of the row to delete.

Return values

None

```
' Example: DeleteRow method
' Creates a 4 X 4 table and deletes the second row.
Dim TableObj As DrawObject
Set TableObj = CurrentPage.CreateTable (1, 4, 4)
TableObj.DeleteRow 2
```

```
' Example: DeleteSpeakerNote method
CurrentPage.DeleteSpeakerNote
```

Freelance Graphics: DeleteSpeakerNote method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DELETESPEAKERNOTE_METHOD_EXSCRIPT',1)} See example

Delete speaker notes from the page.

Syntax

pageobject.**DeleteSpeakerNote**

Parameters

None

Return values

None

```
' Example: Deselect method
CurrentDocument.Deselect
```

Freelance Graphics: Deselect method
{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes
{button ,AL(`H_FLG_DESELECT_METHOD_EXSCRIPT',1)} See example
Clear all the selected objects on the current page.

Syntax
documentobject.**Deselect**

Parameters
None

Return values
None

Freelance Graphics: DistributeForComment method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DISTRIBUTEFORCOMMENT_METHOD_EXSCRIPT',1)} See example

Open the Distribute for Comment dialog box.

Syntax

documentobject.**DistributeForComment**

Parameters

None

Return values

None

```
' Example: DistributeForComment method
' Opens the distribute for comment dialog for the current document.
Dim DocObj As Document
Set DocObj = CurrentDocument
DocObj.DistributeForComment
```

```
' Example: DoVerb method
' In this example, you create an instance of an OLE object
' (a Paintbrush picture), and then using the reference, MyOLE, you
' use the DoVerb method to act on the created object.
Set MyOLE = CurrentPage.CreateObject ("PaintBrush Picture")
EDITPICT = 0
MyOLE.DoVerb EDITPICT
```

Freelance Graphics: DoVerb method

{button ,AL(`;H_FLG_OLEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_DOVERB_METHOD_EXSCRIPT',1)} See example

DoVerb is a variant.

Syntax

OLEObject.**DoVerb***(Index*)

Parameters

index as Integer

Indicates which action to take. Starts with zero. Indicates whatever action is available for this instance of the OLEObject class. For example, a Freelance Graphics presentation OLE object has the actions Edit presentation, Run Screen Show, and Convert. These actions correspond to the index values of 0, 1, and 2, respectively.

Return values

None.

Usage

You use this method to perform actions on an instance of the OLEObject class.

Freelance Graphics: EnterEditMode method

{button ,AL(`;H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ENTEREDITMODE_METHOD_EXSCRIPT',1)} See example

Enter edit mode on a text block.

Syntax

textblockobject.**EnterEditMode**

Parameters

None

Return values

None

```
' Example: EnterEditMode method
' Create a text block and put it in edit mode.
Dim TextObj As DrawObject
Set TextObj = CurrentPage.CreateText
TextObj.EnterEditMode
```

Freelance Graphics: Export method

{button ,AL(`H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_EXPORT_METHOD_EXSCRIPT',1)} See example

Exports the current document or page, as appropriate, to non-Freelance file formats as listed under the FileName parameter description.

Note If you want to save a presentation in HTML format (*.HTM), use the PublishToWeb method.

Syntax

*documentobject.***Export** *(FileName)*

Parameters

FileName As String

Name of file to export current document or page to. FileName should use the standard extension of the file type you wish to create. For example, if you intend to export to a Windows BMP file, the FileName parameter should be a file name with a .BMP extension.

Supported File types & Extensions:

Adobe Illustrator = filename.AI

Windows/PM   BMP = filename.BMP

Computer Graphics Metafile = filename.CGM

Encapsulated PostScript = filename.EPS

JPEG encoded bitmap = filename.JPG

OS/2 Metafile = filename.MET

ZSoft PC Paintbrush Bitmap = filename.PCX

Targa Bitmap = filename.TGA

Tag Image = filename.TIF

WordPerfect Graphic = filename.WPG

Windows Metafile = filename.WMF

Note If there is no extension or if you use an extension that is not one of those listed, the method exits and does not export the file.

Return values

None

Usage

If a directory (folder) is provided with the the file name, it will be created. If no directory is provided, the file will be exported to your working directory (the default is LOTUS\Working\FLG).   If the directory cannot be created (bad directory name or access problems), the method exists and the file is not exported.

```
' Example: Export method
' To save the current document (in most cases,
' this will mean mean the current page) in all available formats:
Sub mysub
   Dim TargetFile As String
   Dim TargetDir As String
   TargetFile = "myfile"
   TargetDir = "c:\testdir\"
   CurrentDocument.Export TargetDir + TargetFile +".ai"
   CurrentDocument.Export TargetDir + TargetFile +".bmp"
   CurrentDocument.Export TargetDir + TargetFile +".cgm"
   CurrentDocument.Export TargetDir + TargetFile +".eps"
   CurrentDocument.Export TargetDir + TargetFile +".met"
   CurrentDocument.Export TargetDir + TargetFile +".pcx"
   CurrentDocument.Export TargetDir + TargetFile +".tga"
   CurrentDocument.Export TargetDir + TargetFile +".tif"
   CurrentDocument.Export TargetDir + TargetFile +".wpg"
   CurrentDocument.Export TargetDir + TargetFile +".wmf"
End Sub
```

```
' Example: FindNextObject method
' Create two rectangles, name them both MyRect, then set Rect3
' to be the same as Rect2, then print the name of Rect3.
Dim Rect1 As DrawObject
Dim Rect2 As DrawObject
Dim Rect3 As DrawObject
Set Rect1 = CurrentPage.CreatRect
Set Rect2 = CurrentPage.CreatRect
Rect1.Name = "MyRect"
Rect2.Name = "MyRect"
Set Rect3 = CurrentPage.FindNextObject("MyRect", Rect1)
Print Rect3.Name
```

Freelance Graphics: FindNextObject method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FINDNEXTOBJECT_METHOD_EXSCRIPT',1)} See example

Find the next occurrence of the specified named object on the Page.

Note Use FindObject to find the first occurrence of an object on a Page.

Syntax

pageobject.**FindNextObject(***objectname, afterme***)**

Parameters

objectname as String

Name of the object to find.

afterme as DrawObject

The object after which to begin the search.

Return values

An instance of the DrawObject class (the found object).

```
' Example: FindObject method
' For this example to work there must be an object on the current
' page named My Rect.
Dim Rect1 As DrawObject
Set Rect1 = CurrentPage.FindObject("My Rect")
```

Freelance Graphics: FindObject method

{button ,AL(`H_FLG_DOCUMENT_CLASS;H_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FINDOBJECT_METHOD_EXSCRIPT',1)} See example

Find the first occurrence of the specified named object in the Document or on the Page.

Note Use FindNextObject to find subsequent occurrences of an object on a Page.

Syntax

object.**FindObject(***objectname***)**

Parameters

objectname as String

Name of the object to find.

Return values

An instance of the DrawObject class.

```
' Example: Flip method
' Flip the selected object from left to right.
Selection.Flip($FlipLeftToRight)
```

Freelance Graphics: Flip method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FLIP_METHOD_EXSCRIPT',1)} See example

Flip the object top-to-bottom or left-to-right.

Syntax

drawobject.**Flip(***flipdirection***)**

Parameters

flipdirection as Variant (Enumerated)

Value	Description
$FlipLeftToRight	Mirror the image on the vertical plane
$FlipTopToBottom	Mirror the image on the horizontal plane

Return values

None

Freelance Graphics: GetBulletCount method

{button ,AL(`;H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETBULLETCOUNT_METHOD_EXSCRIPT',1)} See example

Return the number of bulleted items in the text block.

Syntax

textblock.**GetBulletCount**

Parameters

None

Return values

Number of bullets (Integer)

```
' Example: GetBulletCount method
' For each text block on the current page,
' the number of bullets is printed to the output window.
Forall obj In CurrentPage.Objects
   If (obj.IsText) Then
      Print obj.GetBulletCount
   End If
End Forall
```

Freelance Graphics: GetCell method

{button ,AL(`;H_FLG_TABLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETCELL_METHOD_EXSCRIPT',1)} See example

Return the specified cell as a drawn object.

Syntax

tableobject.**GetCell(***row*, *column***)**

Parameters

row as Integer

Row number of the row containing the text.

column as Integer

Column number of the column containing the text.

Return values

An instance of the TextBlock class.

```
' Example: GetCell method
' Creates a 4 X 4 table and adds text to cell (2, 3).
Dim TableObj As DrawObject
Dim TableCell As TextBlock
Set TableObj = CurrentPage.CreateTable (1, 4, 4)
Set TableCell = TableObj.GetCell(2, 3)
TableCell.Text = "This text is put in the cell."
```

```
' Example: GetEnum method
CurrentApplication.GetEnum("ViewDraw")
```

Freelance Graphics: GetEnum method

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETENUM_METHOD_EXSCRIPT',1)} See example

Resolve the specified enumerated type to its value. This is useful for cross-application scripts.

Syntax

applicationobject.**GetEnum(***enumerationname***)**

Parameters

enumerationname as Variant

An enumeration name as a string, with or without the $.

Return values

Value of the specified enumeration name.

```
' Example: GetIndex method
' There are two examples.
' In the first example, you must have an object variable MyRect
' on the page.
Dim ThisIndex As Integer
ThisIndex = CurrentPage.Objects.GetIndex(MyRect)
' or, next example:
' In this example you must have already identified a color as MyColor.
Dim ThisIndex As Index
ThisIndex = CurrentApplication.Colors.GetIndex(MyColor)
```

Freelance Graphics: GetIndex method

{button ,AL(`H_FLG_COLORS_CLASS;H_DOCUMENTS_CLASS;H_OBJECTS_CLASS;H_PAGES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETINDEX_METHOD_EXSCRIPT',1)} See example

Returns the index (for example, the fifth object in the set).

Syntax

objectset.**GetIndex(***object*, *startingindex***)**

Parameters

object as Object

The instance of the object whose index you want.

startingindex as Integer

Optional starting position in the set for the search.

Return values

Integer indicating the object occurrence within the set of objects (for example, the page number of a specified page within the set of pages).

```
' Example: GetMarkup method
' Creates a text block, underlines it, and
' prints its markup attributes to the output window.
Dim TxtObj As DrawObject
Set TxtObj = CurrentPage.CreateText
TxtObj.Text = "Some text"
TxtObj.Font.Underline = True
Print TxtObj.GetMarkup
```

Freelance Graphics: GetMarkup method

{button ,AL(`;H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETMARKUP_METHOD_EXSCRIPT',1)} See example

Return the text and all markup attributes from a TextBlock, as a string.

Syntax

textblockobject.**GetMarkup**

Parameters

None

Return values

A string containing the text from a TextBlock object, and all markup attributes for that text.

```
' Example: GetNearestColor method
Dim MyLibraryColor As Color
Dim MyColor As Color
Set MyColor = CurrentApplication.Colors.RGBToColor(17395023)
Set MyLibraryColor = CurrentApplication.Colors.GetNearestColor(MyColor)
```

Freelance Graphics: GetNearestColor method

{button ,AL(`H_FLG_COLORS_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETNEARESTCOLOR_METHOD_EXSCRIPT',1)} See example

Return the closest available color (in the Freelance Color Library) to the color specified.

Syntax

*colorsobject***.GetNearestColor(***colorobject***)**

Parameters

colorobject as Color

The Color object whose color you want to match as closely as possible.

Return values

An object of the Color class that is the closest match to the passed color.

Freelance Graphics: GetNearestIndex method

{button ,AL(`H_FLG_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETNEARESTINDEX_METHOD_EXSCRIPT',1)} See example

Return the index of the closest available color for the color specified.

Syntax

colorobject.**GetNearestIndex(***colorobject***)**

Parameters

colorobject as Color

The Color object whose color you want to match as closely as possible.

Return values

An integer containing the index of the Color class that is the closest match to the passed color.

```
' Example: GetNearestIndex method
' Sets the RGBtoColor value for ColorObj and
' finds the nearest color index.
Dim ColorObj As Color
Set ColorObj = CurrentApplication.colors.RGBtoColor(9868800)
Print ColorObj.GetNearestIndex()
```

Freelance Graphics: GetNthBullet method

{button ,AL(`H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETNTHBULLET_METHOD_EXSCRIPT',1)} See example

Return the text from the nth bulleted item within the text block.

Syntax

textblockobject.**GetNthBullet(***n***)**

Parameters

n as Integer

Sequence number of the bullet.

Return values

A string containing the text from the nth bulleted item.

```
' Example: GetNthBullet method
' For each text block on the current page,
' print the text from the second bullet to the output window.
Forall obj In CurrentPage.Objects
   If (obj.IsText) Then
      Print obj.GetNthBullet (2)
   End If
End Forall
```

Freelance Graphics: GetObjectData method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETOBJECTDATA_METHOD_EXSCRIPT',1)} See example

Return user-defined object data (persistent, string-valued name and value pair).

Syntax

drawobject.**GetObjectData(***variablename***)**

Parameters

variablename as String

Name of a user-defined variable.

Return values

String containing the value of the named variable.

```
' Example: GetObjectData method
' This example prints the value of setKey that
' is now associated with the rectangle.
Dim setKey As String
Dim getVal As String
Dim Rectangle As DrawObject
Set Rectangle = CurrentPage.CreateRect
getVal = "value of text"
'Set the value of getVal into setKey.
Call Rectangle.SetObjectData(setKey,getVal)
' Get the value of setKey, then print it.
getVal = Rectangle.GetObjectData(setKey)
Print getVal
```

```
' Example: GetRGB method
Dim RGBValue As Long
RGBValue = MyColor.GetRGB()
```

Freelance Graphics: GetRGB method

{button ,AL(`H_FLG_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETRGB_METHOD_EXSCRIPT',1)} See example

Return the RGB value of the color produced by the current values of the Red, Green, and Blue properties of a Color object.

Syntax

colorobject.**GetRGB**

Parameters

None

Return values

Long, RGB value.

Freelance Graphics: GetSelection method

{button ,AL(`;H_FLG_PAGESELECTION_CLASS;H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETSELECTION_METHOD_EXSCRIPT',1)} See example

Returns the specified selected object from the set of currently selected objects.

Syntax

objectselection.**GetSelection(***objectnumber***)**

Parameters

objectnumber as Integer

The number of the object or page to be returned.

Return values

The DrawObject at the specified index in the selection.

```
' Example: GetSelection method
' You must have at least two draw objects selected on
' the current page for this script to work.  The script
' gets the second of the selected objects and sets
' Rect2 equal to it.
Dim Rect2 As DrawObject
Set Rect2 = Selection.GetSelection(2)
```

Freelance Graphics: GetSpeakerNoteMarkup method
{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GETSPEAKERNOTEMARKUP_METHOD_EXSCRIPT',1)} See example

Return the text from the speaker notes from a page, including all markups.

Syntax
pageobject.**GetSpeakerNoteMarkup**

Parameters
None

Return values
A string containing the text from the speaker notes for a page, and all markup attributes for that text.

```
' Example: GetSpeakerNoteMarkup method
' Gets contents of the speaker note for the current
' page including attributes.  This assumes there
' is already a speaker note for the current page.
Print CurrentPage.GetSpeakerNoteMarkup
```

```
' Example: GotoNotes method
' This assumes that the Freelance Graphics
' presentation is embedded in a Lotus Notes memo
' or document.
Sub Sub1
   CurrentApplicationWindow.GotoNotes
End Sub
```

Freelance Graphics: GotoNotes method

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GOTONOTES_METHOD_EXSCRIPT',1)} See example

Switch window focus to Lotus Notes.

Note This method only works for presentations that are embeded in a Notes document. It is the equivalent of choosing "<Freelance Graphics object> in Lotus Notes" from the Freelance Graphics Windows menu; where <Freelance Graphics object> is information dependent on the state of the Notes document that has Freelance Graphics embedded in it, for example, "New Memo in Lotus Notes."

Syntax

applicationwindow.**GotoNotes**

Parameters

None

Return values

None

Freelance Graphics: GotoPage method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GOTOPAGE_METHOD_EXSCRIPT',1)} See example

Go to the page specified by a page number, page name, or object name.

Syntax

documentobject.**GotoPage** *pageindicator*

Parameters

pageindicator as Variant

Page number (integer), page name (string), or Page object.

Return values

None

```
' Example: GotoPage method
CurrentDocument.GotoPage(5)
```

```
' Example: Group method
Dim MyGroup As DrawObject
Set MyGroup = Selection.Group()
```

Freelance Graphics: Group method

{button ,AL(`;H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GROUP_METHOD_EXSCRIPT',1)} See example

Group currently selected objects.

Syntax

selectionobject.**Group**

Parameters

None

Return values

A new instance of a grouped DrawObject.

```
' Example: Import method
' Imports a cgm file and ignores the other parameters.
Dim DocObj As Document
Set DocObj = CurrentDocument
DocObj.Import "T:\lcam\flg\data\smok\smok.cgm", False, False, False, 0
```

Freelance Graphics: Import method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_IMPORT_METHOD_EXSCRIPT',1)} See example

Import a file.

Syntax

documentobject.**Import(***filename*, *includewithfile*, *converttopostscript*, *includetemplate*, *codepage***)**

Parameters

filename as String

Name of the file to import.

includewithfile as Integer (Enumerated)

Applies to BMP,JPEG,GIF, TIFF, TGA, and Kodak PhotoCD (PCD). If TRUE, the image data is stored directly in the Freelance Presentation file. If FALSE, a link (path and file name) to the image file is stored--the image file is re-imported each time the presentation is opened.

Value	**Description**
TRUE (-1)	Include the imported file with the presentation
FALSE (0)	Do not include the imported file (default)

converttopostscript as Integer (Enumerated)

Applies to Encapsulated Postscript files (EPS) or Adobe Illustrator files (AI). If this parameter is false, the imported image is treated as a Freelance Graphics grouped object.

Value	**Description**
TRUE (-1)	Convert the imported file to PostScript
FALSE (0)	Do not convert the imported file (default)

includetemplate as Integer (Enumerated)

Applies to Computer Graphics Metafile (CGM), Micrografx Draw (DRW), Autocad (DXF), Hewlett-Packard Graphics Language (HGL), Macintosh PICT, Autoshade Rendering (RND), Hewlett-Packard Gallery (GAL), OS/2 Metafile (MET) and WordPerfect Graphics (WPG) files. If true, the background template that is specified in the imported file will be used for the background of the presentation page that contains the imported image.   If includeTemplate is false, then the background of the presentation page will stay the same

Value	**Description**
TRUE (-1)	Include the imported template with the presentation
FALSE (0)	Do not include the imported template (default)

codepage as Integer

Identifies a special code page for the import operation (defaults to 0). For importing text files.

Return values

The imported object.

```
' Example: InsertCol method
' Creates a 4 X 4 table and inserts a column at column number 1.
Dim TableObj As DrawObject
Set TableObj = CurrentPage.CreateTable (1, 4, 4)
TableObj.InsertCol(1)
```

Freelance Graphics: InsertCol method

{button ,AL(`H_FLG_TABLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_INSERTCOL_METHOD_EXSCRIPT',1)} See example

Insert a new column as the specified column in a table.

Syntax

tableobject.**InsertCol(***newcolumnnumber***)**

Parameters

newcolumnnumber as Integer

The column number of the new column in the table.

Return values

None

```
' Example: InsertRow method
' Creates a 4 X 4 table and inserts a row at row number 1.
Dim TableObj As DrawObject
Set TableObj = CurrentPage.CreateTable (1, 4, 4)
TableObj.InsertRow(1)
```

Freelance Graphics: InsertRow method

{button ,AL(`H_FLG_TABLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_INSERTROW_METHOD_EXSCRIPT',1)} See example

Insert a new row as the specified row in a table.

Syntax

tableobject.**InsertRow(***newrownumber***)**

Parameters

newrownumber as Integer

The row number of the new row in the table.

Return values

None

Freelance Graphics: Insert method
{button ,AL(`;H_FLG_PLACEMENTBLOCK_CLASS;',0)} See list of classes
{button ,AL(`H_FLG_INSERT_METHOD_EXSCRIPT',1)} See example
Insert the specified object in a placement ("Click here") block.

Syntax
placmentblockobject.**Insert***(object)*

Parameters
object as DrawObject

The object to be inserted in the "Click here" block.

Return values
None

```
' Example: Insert method
' Creates a rectangle and inserts it into all placement blocks
' on the page.
Dim MyPB As DrawObject
Dim Rectangle As DrawObject
Set Rectangle = CurrentPage.CreateRect
Forall obj In CurrentPage.objects
   If obj.IsPlacementBlock Then
      Set MyPB = obj
      MyPB.PlacementBlock.Insert Rectangle
   End If
End Forall
```

```
' Example: IsEmpty method
If CurrentPage.Objects.IsEmpty() Then
   Print "There are no objects on this page"
End If
```

Freelance Graphics: IsEmpty method

{button ,AL(`H_FLG_COLORS_CLASS;H_FLG_DOCUMENTS_CLASS;H_FLG_OBJECTS_CLASS;H_FLG_PAGES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISEMPTY_METHOD_EXSCRIPT',1)} See example

Test if a collection type object (Colors, Documents, Objects, or Pages) is empty (contains no items).

Syntax

objectcollection.**IsEmpty**

Parameters

None

Return values

Integer (Boolean)

Value	Description
TRUE	There are no items in the object
FALSE	There are one or more items in the object

Freelance Graphics: Item method

{button ,AL(`H_FLG_COLORS_CLASS;H_FLG_DOCUMENTS_CLASS;H_FLG_OBJECTS_CLASS;H_FLG_PAGES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ITEM_METHOD_EXSCRIPT',1)} See example

Return an item from a collection type object (Colors, Documents, Objects, or Pages).

Syntax

objectcollection.**Item(***index***)**

Parameters

index as Variant

The index of the item to return, or the name of the color.

Return values

An instance of the Color, Document, DrawObject, or Page class.

```
' Example: Item method
' There are two examples.
Dim MyPage2 As Page
Set MyPage2 = CurrentDocument.Pages.Item(2)
' or, next example:
Dim MyBlueColor As Color
Dim MyGreenColor As Color
Set MyBlueColor = CurrentApplication.Colors.Item("Blue")
Set MyGreenColor = CurrentApplication.Colors.Item(35)
```

```
' Example: LeaveEditMode method
' Creates a text block and takes it out of edit mode
' after filling it with text.
Dim TextObj As DrawObject
Set TextObj = Currentpage.CreateText
TextObj.EnterEditMode
TextObj.Text = "Some text"
TextObj.LeaveEditMode
```

Freelance Graphics: LeaveEditMode method

{button ,AL(`;H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_LEAVEEDITMODE_METHOD_EXSCRIPT',1)} See example

Leave edit mode for the text block (simulate clicking outside that block).

Syntax

textblockobject.**LeaveEditMode**

Parameters

None

Return values

None

Freelance Graphics: Maximize method

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_MAXIMIZE_METHOD_EXSCRIPT',1)} See example

Maximize the Freelance application or document window.

Syntax

windowobject.**Maximize**

Parameters

None

Return values

None

```
' Example: Maximize method
' There are two examples.
CurrentApplicationWindow.Maximize
' or, next example:
CurrentDocWindow.Maximize
```

Freelance Graphics: Minimize method

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_MINIMIZE_METHOD_EXSCRIPT',1)} See example

Minimize the Freelance application or document window.

Syntax

windowobject.**Minimize**

Parameters

None

Return values

None

```
' Example: Minimize method
' There are two examples.
CurrentApplicationWindow.Minimize
' or, next example:
CurrentDocWindow.Minimize
```

Freelance Graphics: Move method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_MOVE_METHOD_EXSCRIPT',1)} See example

Move an object to a new position on the page, or move a page to or before the specified page number.

Syntax

drawobject.**Move** *xtwips*, *ytwips*

or

pageobject.**Move** *pagenumber, insertbefore*

Parameters

xtwips as Long

Horizontal twips to move (positive moves to right, negative to left).

ytwips as Long

Vertical twips to move (positive moves up, negative down).

pagenumber as Integer

Page number of the new page, or page before which to insert the new page (see note).

insertbefore as Integer (Enumerated)

(Optional)

Value	Description
TRUE (-1)	Insert the page before the specified page number
FALSE (0)	(Default) Insert the page at the specified page number

Return values

None

```
' Example: Move method
' Creates an ellipse and moves it to coordinates 1500, 2000.
Dim Ellipse As DrawObject
Set Ellipse = CurrentPage.CreateOval
Ellipse.Move 1500, 2000
```

```
' Example: NearestColorFromRGB method
Dim MyColor As Color
Set MyColor = CurrentApplication.NearestColorFromRGB(1598564)
```

Freelance Graphics: NearestColorFromRGB method

{button ,AL(`;H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_NEARESTCOLORFROMRGB_METHOD_EXSCRIPT',1)} See example

Return the Freelance Graphics palette color that is the closest match to the specified RGB value.

Syntax

applicationobject.**NearestColorFromRGB(***rgbvalue***)**

Parameters

rgbvalue as Long

Any RGB value.

Return values

An instance of the Color class.

```
' Example: NewDocument method
'Creates a new document with default settings within Freelance.
CurrentApplication.NewDocument
```

Freelance Graphics: NewDocument method

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_NEWDOCUMENT_METHOD_EXSCRIPT',1)} See example

Create a new presentation by showing the user interface, with optional parameters supplying default values or selections.

Syntax

applicationobject.**NewDocument(***name*, *location*, *kind*, *mastername*, *masterlocation)*

Parameters

Note The first three parameters are ignored. You must save a document to name it.

name as String

Optional file name.

location as Variant

Optional location of the file.

kind as String

Optional file type.

mastername as String

Optional content topic (.SMC) or SmartLook (.MAS) file name.

masterlocation as Variant

Optional content topic set location (must contain a string).

Return values

Returns the new document as an instance of the Document class.

Freelance Graphics: OpenDocumentFromInternet method

{button ,AL(`;H_FLG_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_FLG_OPENDOCUMENTFROMINTERNET_METHOD_EXSCRIPT',1)} See example

Opens an existing .PRZ, .PRE, .SMC, .MAS, .SYM, .DGM, or .PAL document from the Internet, analogous to the OpenDocument function. Returns the document.

Syntax

*applicationobject***.OpenDocumentFromInternet***(URL, doctype, password, makevisible, userid, userpassword, usepassiveconnection, proxyserveraddress, proxyport, proxytype)*

Parameters

URL As String

Optional. Universal Resource Language location (for example, "ftp://Radium/users/bob/test.123"), if omitted the interactive dialog will appear to prompt the user.

doctype As String

Optional.

password As String

Optional.

makevisible As Integer

Optional (Freelance Graphics ignores this parameter). Integer used as a boolean: Any non-zero integer is true. False is zero.

userid As String

Optional. The ID required by the host.

userpassword As String

Optional. The password required for the host.

usepassiveconnection As Integer

Optional. Integer used as a boolean. True (any non-zero integer is true): your internal network is connected to the Internet through a firewall that supports passive transfers. False (that is, zero): the fire wall does not accept passive transfers.

proxyserveraddress As String

Optional. Server IP Address or Domain Name

proxyport As Integer

Optional. Choose from the available proxy ports.

proxytype As Integer

Optional. A World Wide Web page is INT_WWW (1) or an FTP file is INT_FTP (2)

Return values

Returns an instance of the Document class.

Usage

Use this method to open a Freelance Graphics file on the internet.

```
' Example: OpenDocumentFromInternet method
' Opens a file from the internet assuming the user is currently
' connected and no id's and passwords need to be given.
' Note that the underscore is used as a line continuation
' character in LotusScript.
CurrentApplication.OpenDocumentFromInternet _
   "ftp://Saturn/users/Jennifer/q1budget.prz"
```

Freelance Graphics: OpenDocumentFromNotes method

{button ,AL(`;H_FLG_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_FLG_OPENDOCUMENTFROMNOTES_METHOD_EXSCRIPT',1)} See example

Opens a document attached to a Notes document. Returns the document.

Syntax

*applicationobject.***OpenDocumentFromNotes**(*attachedfilename, universalnotesid, fieldname, databasepath, servername, doctype, password, openasreadonly, makevisible)*

Parameters

attachedfilename As String

For example, a file name such as"test.123."   If omitted the interactive dialog will appear to prompt the user. If you specify this parameter, you must either specify the UniversalNotesID parameter or the DatabasePath, ServerName, and Password (if necessary) parameters.

universalnotesid As String

Optional, but see attachedfilename parameter. For example, "150DFE45F1089B790065828D852562CA"

fieldname As String

Optional. For example, "Body"

databasepath As String

Optional but see attachedfilename parameter. For example, "Databases\Docs in Progress.nsf"

servername As String

Optional. For example, "Local"

doctype As String

Optional but see attachedfilename parameter. For example, ".PRZ"

password As String

Optional but see attachedfilename parameter. Password of document to open.

openasreadonly As Integer

Optional (Freelance Graphics ignores this parameter). Integer used as a boolean: Any non-zero integer is true. False is zero.

MakeVisible As Integer

Optional (Freelance Graphics ignores this parameter). Integer used as a boolean: Any non-zero integer is true. False is zero.

Return values

Returns an instance of the Document class.

Usage

Use this method to open a presentation that is embedded a Notes docment.

```
' Example: OpenDocumentFromNotes method
' Note that the underscore is used as a line continuation
' character in LotusScript.
CurrentApplication.OpenDocumentFromNotes "test.prz", "somenotesid", _
          "Body", "Progress.nsf", "Local"
```

```
' Example: OpenDocument method
' Opens a prz file and accepts the default values for
' the optional parameters.
' Passwords need to be given.
CurrentApplication.OpenDocument "c:\lotus\work\flg\pres1.prz"
```

Freelance Graphics: OpenDocument method

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_OPENDOCUMENT_METHOD_EXSCRIPT',1)} See example

Open a presentation for editing.

Syntax

applicationobject.**OpenDocument(***name*, *location*, *kind*, *readonly*, *makevisible)*

Parameters

name As String

File name of the presentation file to be opened.

location As Variant

Optional path for the file (must contain a string; defaults to the working directory).

kind As String

Optional file type (defaults to PRZ).

readonly As Integer (Enumerated)

Optional read-only flag.

Value	Description
TRUE (-1)	Read-only
FALSE (0)	Read-write (default)

makevisible As Integer (Enumerated)

Ignored and always treated as TRUE (this parameter is declared for compatibility with Open methods in other Lotus products).

Value	Description
TRUE (-1)	Visible (default)
FALSE (0)	Invisible (ignored)

Return values

Returns the opened document as an instance of the Document class.

Freelance Graphics: OpenPresForCopy method
{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_OPENPRESFORCOPY_METHOD_EXSCRIPT',1)} See example

Open a presentation from which pages can be copied (similar to the presentation browser; however, this method does not open or use the presentation browser).

Syntax
documentobject.**OpenPresForCopy(***presentationname***)**

Parameters
presentationname as String

Name of the presentation file to open in the browser.

Return values
None

Usage
Use this method only in conjunction with two other methods: SelectPageForCopy method and PasteSelectedPages method.

```
' Example: OpenPresForCopy method
Sub CopyAPage
   CurrentDocument.OpenPresForCopy "turtle.prz"
   CurrentDocument.SelectPageForCopy 1
   CurrentDocument.PasteSelectedPages 1
End Sub
```

```
' Example: PastePage method
'Creates a document, copies page 1 and pastes it at page 3.
Dim DocObj As Document
Set DocObj = CurrentDocument
DocObj.GotoPage 1
CurrentPage.CopyPage
DocObj.GotoPage 3
DocObj.PastePage
```

Freelance Graphics: PastePage method

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_PASTEPAGE_METHOD_EXSCRIPT',1)} See example

Pastes a page into the current document after the current page.

Syntax

documentobject.**PastePage**

Parameters

None

Return values

Returns a Page object.

Usage

Before calling this method, a Freelance Graphics page must have been placed on the clipboard by a user or by a script command.

Freelance Graphics: PasteSelectedPages method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PASTESELECTEDPAGES_METHOD_EXSCRIPT',1)} See example

Paste into the document the page(s) copied by the SelectPageForCopy method.

Syntax

documentobject.**PasteSelectedPages(***pagestartnum***)**

Parameters

pagestartnum as Integer

Where the selected pages are to be pasted.

Value	Description
0	After the current page
1	Before the current page
2	At the end of the presentation

Return values

None

```
' Example: PasteSelectedPages method
' Opens a file, selects a page to copy into the current document, and
' pastes it in the current document.
CurrentDocument.OpenPresForCopy "c:\lotus\work\flg\ls.prz"
CurrentDocument.SelectPageForCopy(1)
CurrentDocument.PasteSelectedPages(0)
```

Freelance Graphics: PasteSpecial method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PASTESPECIAL_METHOD_EXSCRIPT',1)} See example

Paste the clipboard contents onto the page using a special format (bitmap or chart, for example).

Syntax

pageobject.**PasteSpecial(***clipboardformat***)**

Parameters

clipboardformat as String

Any of the clipboard formats displayed in the Paste Special dialog box, or *OLE object* for an OLE object on the clipboard.

Return values

The pasted object (a new instance of the DrawObject class).

```
' Example: PasteSpecial method
'Creates a rectangle, copies it, and pastes it as a bitmap.
Dim Rectobject As DrawObject
Set RectObject = CurrentPage.CreateRect
RectObject.Copy
CurrentPage.PasteSpecial("Bitmap")
```

Freelance Graphics: Paste method

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PASTE_METHOD_EXSCRIPT',1)} See example

Paste the Clipboard contents onto the page or into the document.

Syntax

object.**Paste(***makevisible***)**

Parameters

makevisible as Integer (Enumerated)

Ignored and always treated as TRUE (this parameter is declared for compatibility with Paste methods in other Lotus products).

Value	Description
TRUE (-1)	Make the pasted page(s) visible (default)
FALSE (0)	Make the pasted page(s) invisible (ignored)

Return values

If pasting to a Page, the pasted object (a new instance of the DrawObject class);   if pasting to a Document, no return value.

```
' Example: Paste method
'Creates a rectangle, copies it, and pastes it.
Dim Rectobject As DrawObject
Set RectObject = CurrentPage.CreateRect
RectObject.Copy
CurrentPage.Paste
```

Freelance Graphics: Play method
{button ,AL(`H_FLG_MEDIA_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PLAY_METHOD_EXSCRIPT',1)} See example

Play a media object (a sound or a movie).

Syntax
mediaobject.**Play**

Parameters
None

Return values
None

```
' Example: Play method
' Adds a movie to the page and plays it.
Dim Movie As DrawObject
Set Movie = CurrentPage.CreateMovie("c:\lotus\flg\media\dinosaur.aim")
Movie.Media.Play
```

Freelance Graphics: PrintOut method

{button ,AL(`;H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PRINTOUT_METHOD_EXSCRIPT',1)} See example

Print the presentation.

Syntax

documentobject.**PrintOut(***frompage*, *topage*, *copies)*

Parameters

frompage as Integer

Starting page number to print.

topage as Integer

Ending page number to print.

copies as Integer

Number of copies to print.

Return values

None

Usage

This method is identical to the Print method.

```
' Example: PrintOut method
' Prints the first page of the current document.
CurrentDocument.PrintOut 1,1, 1
```

```
' Example: Print method
' Prints the first page of the current document.
CurrentDocument.Print 1,1, 1
```

Freelance Graphics: Print method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PRINT_METHOD_EXSCRIPT',1)} See example

Print the presentation.

Syntax

documentobject.**Print(***frompage***,** *topage***,** *copies)*

Parameters

frompage as Integer

Starting page number to print.

topage as Integer

Ending page number to print.

copies as Integer

Number of copies to print.

Return values

None

Usage

This method is identical to the PrintOut method.

```
' Example: PublishToWeb method
' Publishes the specified file to the web.
CurrentDocument.PublishToWeb "Test", $NonHTML20, True
```

Freelance Graphics: PublishToWeb method

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLGPUBLISHTOWEB_METHOD_EXSCRIPT',1)} See example

Saves a presentation in HTML format (including GIF files, content page, link to presentation, speaker notes, and so on).

Syntax

DocumentObject.**PublishToWeb**(*FileName, FileType, Format, Resolution, Contents, LinkPres, SpkNote, Media, Email, EmailName, EmailAddr,ImageType)*

Parameters

FileName As String

Name of main HTM file. If directory is not specified, uses work directory. Appends ".HTM", if not specified.

FileType As Variant

Optional parameter.

$NonHTML20 = Not HTML 2.0 (Default)

$HTML20CERN = HTML 2.0 CERN

$HTML20NCSA = HTML 2.0 NCSA

Format As Integer

Optional parameter.

0 or FALSE = HMTL 2.0

1 or TRUE = Single image per page (default)

2 = Separated images and text

3 = Plug-in

4 = ActiveX component

Resolution As Variant

Optional parameter.

$Res640x480

$Res800x600 (Default)

$Res1024x768

$Res1280x1024

Contents As Integer

Optional parameter.

FALSE = no table of contents

TRUE = table of contents (Default)

LinkPres As Integer

Optional parameter.

FALSE = no link to PRZ file (Default)

TRUE = add link

LotusLink As Integer

Optional parameter.

FALSE = no button linked to Lotus Home Page

TRUE = add button linked to Lotus Home Page (Default)

SpkNote As Integer

Optional parameter.

FALSE = no speaker notes

TRUE = publish speaker notes (Default)

Media As Integer

Optional parameter.

FALSE = don't publish multimedia data (Default)

TRUE = publish multimedia data

Email As Integer

Optional parameter. If this parameter is TRUE, then EmailName and EmailAddr parameters must be included.

FALSE = no mail to URL (Default)

TRUE = add mail to URL

EmailName As String

Optional parameter. However, if Email is True, this parameter must be included.

String for name in mail to URL.

EmailAddr As String

Optional parameter. However, if Email is True, this parameter must be included.

String form email address in mail to URL.

ImageType As Variant

Optional parameter (prefered image type).

$GIF = GIF format (best for graphs, charts, and solid colors)

$JPEG = JPEG format(best for photographs and gradfills)

Return values

None.

Usage

You can use PublishToWeb to save a presentation in HTML format.

Freelance Graphics: PutIntoPlacementBlock method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PUTINTOPLACEMENTBLOCK_METHOD_EXSCRIPT',1)} See example

Place an object in a "Click here" block.

Syntax

drawobject.**PutIntoPlacementBlock(***placementblockid***)**

Parameters

placementblockid as Integer

An integer identifying the "Click here" block.

Return values

None

```
' Example: PutIntoPlacementBlock method
' Creates a rectangle and puts it in the first placement block
' on the page.
Dim Rectangle As DrawObject
Set Rectangle = CurrentPage.CreateRect
Rectangle.PutIntoPlacementBlock(1)
```

```
' Example: Quit method
'Exits Freelance Graphics.
CurrentApplication.Quit
```

Freelance Graphics: Quit method

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_QUIT_METHOD_EXSCRIPT',1)} See example

Exit Freelance Graphics.

Syntax

applicationobject.**Quit**

Parameters

None

Return values

None

```
' Example: RemoveFromSelection method
Selection.RemoveFromSelection MyRect2
```

Freelance Graphics: RemoveFromSelection method

{button ,AL(`H_FLG_DOCUMENT_CLASS;H_FLG_PAGESELECTION_CLASS;H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_REMOVEFROMSELECTION_METHOD_EXSCRIPT',1)} See example

Remove the specified object from the set of currently selected objects.

Syntax

objectcollection.**RemoveFromSelection(***object)*

Parameters

object as DrawObject

The object you want removed from the selection.

Return values

None

Freelance Graphics: Remove method
{button ,AL(`H_FLG_DRAWOBJECT_CLASS;H_FLG_PAGE_CLASS;',0)} See list of classes
{button ,AL(`H_FLG_REMOVE_METHOD_EXSCRIPT',1)} See example
Delete a DrawObject or Page object.

Syntax
object.**Remove**

Parameters
None

Return values
None

```
' Example: Remove method
Selection.Remove
```

Freelance Graphics: Replicate method
{button ,AL(`H_FLG_DRAWOBJECT_CLASS;H_FLG_PAGE_CLASS;',0)} See list of classes
{button ,AL(`H_FLG_REPLICATE_METHOD_EXSCRIPT',1)} See example
Replicate an object.

Syntax
drawobject.**Replicate**

Parameters
None

Return values
A new instance of the replicated object.

```
' Example: Replicate method
' Creates a rectangle and replicates it.
Dim Rectangle As DrawObject
Set Rectangle = CurrentPage.CreateRect
Rectangle.Replicate
```

```
' Example: Restore method
' Gives the Freelance application window the focus.
CurrentApplicationWindow.Restore
```

Freelance Graphics: Restore method

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_RESTORE_METHOD_EXSCRIPT',1)} See example

Restore the Freelance Graphics application or document window.

Syntax

windowobject.**Restore**

Parameters

None

Return values

None

Freelance Graphics: RevertToStyle method

{button ,AL(`H_FLG_BACKGROUND_CLASS;H_FLG_BORDER_CLASS;H_FLG_FONT_CLASS;H_FLG_LINESTYLE_CLASS;H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_REVERTTOSTYLE_METHOD_EXSCRIPT',1)} See example

Revert to the named style for a Background, Border, Font, LineStyle, or TextBlock object.

Note For this release, the RevertToStyle method has no effect any object class.

Syntax

object.**RevertToStyle(***attributename***)**

Parameters

attributename as String

Optional input parameter.

Return values

None

```
' Example: RevertToStyle method
' Creates a text block, changes its attributes, and reverts to style.
Dim TextObj As DrawObject
Set TextObj = Currentpage.CreateText
TextObj.Text = "Some text"
TextObj.TextBlock.Font.Underline = True
Textobj.RevertToStyle
```

```
' Example: RGBtoColor method
Dim MyColor As Color
Set MyColor = CurrentApplication.Colors.RGBToColor(1598564)
```

Freelance Graphics: RGBtoColor method
{button ,AL(`H_FLG_COLORS_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_RGBTOCOLOR_METHOD_EXSCRIPT',1)} See example

Return the Color class object that matches the RGB value specified.

Syntax
colorsobject.**RGBtoColor(***colornumber***)**

Parameters
colornumber as Long

An RGB value.

Return values
Color class object matching the RGB value specified.

Freelance Graphics: Rotate method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ROTATE_METHOD_EXSCRIPT',1)} See example

Rotate the object a specified number of degrees around a specified anchor point.

Syntax

drawobject.**Rotate** *xanchor*, *yanchor*, *degrees*

Parameters

xanchor as Long

Horizontal coordinate of the anchor point in twips.

yanchor as Long

Vertical coordinate of the anchor point in twips.

degrees as Double

Degrees of rotation in a counterclockwise direction.

Return values

None

```
' Example: Rotate method
' Creates a rectangle and rotates it 25 degrees.
Dim Rectangle As DrawObject
Set Rectangle = CurrentPage.CreateRect (1500, 2000, 1500, 2000)
Rectangle.Rotate 1500, 2000, 25
```

```
' Example: RunDialog method
' Example of posting Dialog 2 and interpreting results.
' Note that the underscore is used as a line continuation
' character in LotusScript.
DIM PackedVal As Integer
PackedVal = CurrentDocument.RunDialog (2, "DialogTitle", "StaticText",_
"CheckBox 1", "CheckBox 2", "CheckBox 3", "CheckBox 4", "CheckBox 5",_
"CheckBox 6", "CheckBox 7", FALSE, 0)
For I = 1 TO 7
    IF PackedVal AND (2^(I-1)) THEN
       Print "CheckBox "+Str$(I)+" was enabled."
    ELSE
        Print "CheckBox "+Str$(I)+" was disabled."
    END IF
NEXT I
```

Freelance Graphics: RunDialog method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_RUNDIALOG_METHOD_EXSCRIPT',1)} See example

Launch one of five hard-coded dialogs.

Syntax

***documentobject*.RunDialog(*dialogtype*, *dialogcaption*, *maintext*, *p4*, *p5*, *p6*, *p7*, *p8*, *p9*, *p10*, *p11*, *p12*)**

Parameters and return values (by dialog type)

The first three parameters are the same for all five dialog types. The usage of parameters 4-12 and the return values vary by dialog type. The common parameters are described first, followed by the dialog-specific parameters and return values for each dialog type.

This method always returns a packed 32-bit (integer) value. In the descriptions of the return values, bit 0 is the least significant bit of the integer.

Common parameters

For all dialogs the first three parameters are always the same.

dialogtype as Integer

Value	Description
1	Two radio buttons, each with an associated spinner control
2	Seven check boxes
3	Four radio buttons, optional second text box
4	Three check boxes, optional second text box
5	List box with up to seven items

dialogcaption as String

The text to display in the caption (title bar) of the dialog.

maintext as String

Instructional text for a display-only text box in the dialog.

The use of the remaining parameters varies depending on the value of first parameter (*dialogtype*).

Note Parameters 4-10 are always type String, although for some dialog types they contain string representations of numerics. Parameters 11 and 12 are always type Integer.

Type 1 (two radio buttons, each with an associated spinner control) parameters

Parameter	Description
p4	String for first radio button
p5	String for second radio button
p6	String containing the number of the radio button selected initially
p7, p8	Unused
p9	String containing the first spinner minimum value
p10	String containing the first spinner maximum value
p11	Integer containing the second spinner minimum value
p12	Integer containing the second spinner maximum value

Type 1 return values

Bit(s)	Description
0-7	The value set by the spinner control for the selected radio button when the dialog was closed
8	The radio button selected on return:

Bit(s)	Description
	0 = first
	1 = second
9	How the user closed the box:
	0 = selected Cancel
	1 = selected OK
10-31	Unused

Type 2 (seven check boxes) parameters

Parameter	Description
p4-p10	Strings for check boxes 1-7; each parameter is the string to display with the corresponding check box
p11	Initial state of the check boxes; True (non-zero) sets all check boxes as checked; False (zero) sets all check boxes as unchecked.
p12	Unused

Type 2 return values

Bit(s)	Description
0-6	Indicates which boxes were selected on return:
	0 = not selected
	1 = selected
7	Unused
8	How the user closed the box:
	0 = selected Cancel
	1 = selected OK
9-31	Unused

Type 3 (four radio buttons, optional second text box) parameters

Parameter	Description
p4-p7	Strings for radio buttons 1-4; each parameter is the string to display with the corresponding radio button
p8	String for optional second text block
p9-p12	Unused

Type 3 return values

Bit(s)	Description
0-2	A binary number indicating which radio button was selected on return (for example, 110 indicates that button three was selected)
3-7	Unused
8	How the user closed the box:
	0 = selected Cancel
	1 = selected OK
9-31	Unused

Type 4 (three check boxes, optional secondary text) parameters

Parameter	Description

p4-p6	Strings for check boxes 1-3; each parameter is the string to display with the corresponding check box
p7	String for optional secondary text block
p8-p10	Unused
p11	Initial state of the check boxes; True (non-zero) sets all check boxes as checked; False (zero) sets all check boxes as unchecked.
p12	Unused

Type 4 return value

Bit(s)	Description
0-2	Indicates which boxes were selected on return:
	0 = not selected
	1 = selected
3-7	Unused
8	How the user closed the box:
	0 = selected Cancel
	1 = selected OK
9-31	Unused

Type 5 (list box with up to seven items) parameters

Parameter	Description
p4-p10	Up to seven strings; each string is a separate entry in the list box
p11-p12	Unused

Type 5 return values

Bit(s)	Description
0-2	A binary number indicating which item in the list box was selected on return (for example, 110 indicates that item three was selected)
3-7	Unused
8	How the user closed the box:
	0 = selected Cancel
	1 = selected OK
9-31	Unused

Freelance Graphics: SameColor method

{button ,AL(`H_FLG_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SAMECOLOR_METHOD_EXSCRIPT',1)} See example

Compare the color of two Color class objects to determine if they have the same RGB value.

Syntax

*colorobject1***.SameColor(***colorobject2***)**

Parameters

colorobject2 as Color

An object of class Color.

Return values

Integer (Boolean).

Value	Description
TRUE	Same color
FALSE	Different colors

```
' Example: SameColor method
If Color1.SameColor(Color2) Then
   Print "Color2 is identical to Color1"
End If
```

Freelance Graphics: SaveAsToInternet method

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SAVEASTOINTERNET_METHOD_EXSCRIPT',1)} See example

Saves the document to the Internet (analogous to the SaveAs method).

Syntax

DocumentObject.**SaveAsToInternet**(*URL, DocType, Password, UserID, UserPassword, UsePassiveConnection, ProxyServerAddress, ProxyPort, ProxyType)*

Parameters

URL as String

Optional. Universal Resource Language location (for example, "ftp://Radium/users/bob/test.123"), if omitted the interactive dialog will appear to prompt the user.

doctype as String

Optional (Freelance Graphics ignores this parameter). For example: ".PRZ"

password as String

Optional. The document password.

userid as String

Optional. The user's ID for the Internet server.

userpassword as String

Optional.

usepassiveconnection as Integer

Optional. Integer used as a boolean. True (any non-zero integer is true): your internal network is connected to the Internet through a firewall that supports passive transfers. False (that is, zero): the firewall does not accept passive transfers.

proxyserveraddress as String

Optional. The server IP Address or Domain Name

proxyport as Integer

Optional.

proxytype as Integer

Optional. For a World Wide Web page, use INT_WWW (1) ; for an FTP file, use INT_FTP (2)

Return values

None.

Usage

Use this method to save files as a given file type to an Internet server.

```
' Example: SaveAsToInternet method
'  Save the current document as an HTM document on the Web site.
CurrentDocument.SaveAsToInternet "http://www.yoursite.com/data/account.prz"
```

Freelance Graphics: SaveAsToNotes method

{button ,AL(`;H_FLG_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SAVEASTONOTES_METHOD_EXSCRIPT',1)} See example

Saves the document as a Notes attachment.

Syntax

documentobject.**SaveAsToNotes**(*attachedfilename, universalnotesid, fieldname, databasepath, servername, doctype, password)*

Parameters

attachedfilename as String

For example: "test.123," if omitted the interactive dialog will appear to prompt the user. If you specify this parameter, you must either specify the UniversalNotesID parameter or the DatabasePath, ServerName, and Password (if necessary) parameters.

universalnotesid as String

Optional but see attachedfilename parameter. For example: "150DFE45F1089B790065828D852562CA"

fieldname as String

Optional, however, if it is not specified it will put the document in the first rich text field it finds. For example, to specify a field type: "Body"

databasepath as String

Optional but see attachedfilename parameter. For example: "Databases\Docs in Progress.nsf"

servername as String

Optional but see attachedfilename parameter. For example: "Local"

doctype as String

Optional (Freelance Graphics ignores this parameter). For example: ".PRZ"

password as String

Optional but see attachedfilename parameter. The document password.

Return values

None.

Usage

Use this method to save presentations to a Notes server.

```
' Example: SaveAsToNotes method
' Save the presentation to a Notes database, and put it in
' the Body field of the document.
' Note that the underscore is used as a line continuation
' character in LotusScript.
CurrentDocument.SaveAsToNotes "filename.prz", _
   82857CD23777850E8525643D00727DCB,"Body"
```

```
' Example: SaveAs method
' Saves the current document to the specified directory and makes a backup.
CurrentDocument.SaveAs "Test.prz", "c:\lotus\work\flg",, True
```

Freelance Graphics: SaveAs method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SAVEAS_METHOD_EXSCRIPT',1)} See example

Save the current presentation in a new file.

Syntax

documentobject.**SaveAs(***filename*, *location*, *type, backup***)**

Parameters

filename as String

Optional file name.

location as Variant

Optional path in which to save the file.

type as String

(Optional) File type is ignored for this release. The actual type is always PRZ.

backup as Integer

Indicates if the file should be backed up.

Value	Description
TRUE (-1)	Back up the file
FALSE (0)	Do not back up the file

Return values

None

Freelance Graphics: Save method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SAVE_METHOD_EXSCRIPT',1)} See example

Save the current presentation.

Syntax

documentobject.**Save**

Parameters

None

Return values

None

```
' Example: Save method
' Saves the current document to the specified directory and makes a backup.
CurrentDocument.Save
```

Freelance Graphics: SelectPageForCopy method
{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SELECTPAGEFORCOPY_METHOD_EXSCRIPT',1)} See example

Select for copying the specified page of a presentation opened with the OpenPresForCopy method.

Syntax
documentobject.**SelectPageForCopy(***pagenumber***)**

Parameters
pagenumber as Integer

The number of the page to select for copying.

Return values
None

```
' Example: SelectPageForCopy method
' Opens a file, selects a page to copy into the current document.
CurrentDocument.OpenPresForCopy "c:\lotus\work\flg\ls.prz"
CurrentDocument.SelectPageForCopy(1)
```

Freelance Graphics: Select method

{button ,AL(`H_FLG_DOCUMENT_CLASS;H_FLG_PAGESELECTION_CLASS;H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SELECT_METHOD_EXSCRIPT',1)} See example

Select the specified document, page, or object.

Syntax

*selectionclasstype***.Select(***object***)**

Parameters

object as objecttype

A Document, DrawObject, or Page object.

Return values

None

```
' Example: Select method
Selection.Select MyRect
```

Freelance Graphics: SetInternetOptions method

{button ,AL(`;H_FLG_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SETINTERNETOPTIONS_METHOD_EXSCRIPT',1)} See example

This method opens the Internet Options dialog box. The user can select any options within the dialog box and then press OK.

Syntax

ApplicationObject.**SetInternetOptions**

Parameters

None.

Return values

None.

Usage

Use this method to connect to a different host or to connect to a host.

```
' Example: SetInternetOptions method
' Opens the internet options dialog box.
CurrentApplication.SetInternetOptions
```

```
' Example: SetObjectData method
' Sets a string value for the rectangle into a variable SetKey.
Dim ob As DrawObject
Set ob = CurrentPage.CreateRect
Dim SetKey As String
Dim SetVal As String
setVal = "Rectangle object data"
Call Ob.SetObjectData(setKey,setVal)
```

Freelance Graphics: SetObjectData method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SETOBJECTDATA_METHOD_EXSCRIPT',1)} See example

Set user-defined object data (persistent, string-valued name and value pair).

Syntax

*drawobject***.SetObjectData(***variablename***,** *variablevalue***)**

Parameters

variablename as String

The name for the variable.

variablevalue as String

The value for the variable.

Return values

None

Freelance Graphics: SetSSAction method

{button ,AL(`;H_FLG_DRAWOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_SETSSACTION_METHOD_EXSCRIPT',1)} See example

Attach "click action" properties to the DrawObjects that support them. These click actions apply when a user clicks the object during the execution of a screenshow. The argument list varies based on the type of action used. The click action and associated properties are passed into SetSSaction as a single semi-colon delimited string.

Syntax

*drawobject.***SetSSAction***(ArgumentList as String)*

Parameters

ArgumentList

A semicolon delimited string containing all necessary settings for a given click action. The first item in the string is the *Actiontype*, followed by additional arguments appropriate to that particular action (if any). The string takes this form: SetSSAction(***Actiontype;arg1;arg2;...;argx***).

Actiontype	Description	Arguments	Description
ssActionNextpage	Go to next page in show when object is clicked.	None	
ssActionPrevpage	Go to previous page in show when object is clicked.	None	
ssActionFirstpage	Go to first page in show when object is clicked.	None	
ssActionLastpage	Go to last page in show when object is clicked.	None	
ssActionLastpagedisplayed	Go to the most recently displayed page in show when object is clicked. For example, if you had just jumped from page 2 to page 6, this action would return you to page 2.	None	
ssActionQuitshow	Exit the screenshow when object is clicked.	None	
ssActionPause	Pause the screenshow when object is clicked. Applies during automatic timed screenshows.	None	
ssActionList	Show dialog containing list of screenshow pages to jump to when object is clicked.	None	
ssActionJumptopage	Jump to the page identified in the pageid argument when object is clicked.	pageid	A valid page name or page number--the page number must be within range of available pages in presentation.

ssActionRunapp	Launch the executable identified in the exename argument when object is clicked.	exename	Full path and file name of an executable program.
ssActionPlaysound	Play a sound when the object is clicked during a screenshow. The movie file and associated properties are defined in the arguments listed here.	filename	Full path and file name of the media file--the file does not need to be available when the script is created, but it must be available at the indicated path when the screenshow is executed, or an error is displayed.
		playcount	Number of times to play the sound or movie. A 0 (zero) indicates that the media should be played continuously.
		embedflag	0 (or ssActionMediaEmbed) - Embed the media file in the presentation. 1 (or ssActionMediaLinked) - Link the media file to the presentation. This setting requires that the media file be available at the indicated location when the screen show runs.
ssActionPlaymovie	Play a movie.	filename	Full path and file name of the media file--the file does not need to be available when the script is created, but it must be available at the indicated path when the screenshow is executed, or an error is displayed.
		playcount	Number of times to play the sound or movie-- 0 (zero) indicates that the media should be played continuously.
		embedflag	0 (or ssActionMediaEmbed) - Embed the media file in the presentation. 1 (or ssActionMediaLinked) - Link the media file to the presentation. This setting requires that the media file at the indicated location when the screen show runs.
		playlocation	0 (or ssActionMediaLocationAtbutton) = Play at location of object clicked upon. 1 (or ssActionMediaLocationLefttop) = Play at left top. 2 (or ssActionMediaLocationTop) = Play at top. 3 (or ssActionMediaLocationRighttop) = Play at right top 4 (or ssActionMediaLocationLeftcenter) = Play at left center. 5 (or ssActionMediaLocationCenter) = Play at center. 6 (or ssActionMediaLocationRightcenter) = Play at right center. 7 (or ssActionMediaLocationLeftbottom) = Play at left bottom. 8 (or ssActionMediaLocationBottom) = Play at bottom. 9 (or ssActionMediaLocationRightbottom) = Play at right bottom.

			Values outside these legal values will be ignored, and movie will play at the default location, or previously set location.
		playspeed	0 (or ssActionMediaPlayspeedDefault) = Default speed. 1 (or ssActionMediaPlayspeedSlow) = Slow. 2 (or ssActionMediaPlayspeedMedium) = Medium. 3 (or ssActionMediaPlayspeedFast) = Fast.
ssActionRunshow	Launch a screenshow based on the Freelance file provided.	filename	Full path and file name of a valid Freelance Graphics (PRZ) file--the file does not need to be available when the script is created, but it must be available at the indicated path when the screenshow is executed, or an error is displayed.
ssActionJumptourl	Launch the default internet browser and jump to the URL indicated.	url	Valid url. The browser, internet connection and/or url do not need to be available when the script is created, but must be available when the screenshow is executed, or an error is displayed.
ssActionNoaction	Sets no click action for object. Removes any existing actions.	None	

Return values

None

```
' Example: SetSSAction method
' An example of each action type is provided.

' ssActionNextpage example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionNextpage")

' ssActionPrevpage example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionPrevpage")

' ssActionFirstpage example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionFirstpage")

' ssActionLastpage example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionLastpage")

' ssActionLastpagedisplayed example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionLastpagedisplayed")

' ssActionQuitshow example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionList")

' ssActionJumptopage example
Dim obj1 as DrawObject
Dim obj2 as DrawObject
Set obj1 = CurrentPage.CreateRect
Set obj2 = CurrentPage.CreateOval
'Clicking obj1 during show jumps to page 2...
obj1.SetSSAction("ssActionJumptopage;2")
'Clicking obj2 during show jumps to page named "Agenda"
obj2.SetSSAction("ssActionJumptopage;Agenda")

' ssActionRunapp example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
' Clicking myobj during show launches calc.exe
```

```
myobj.SetSSAction("ssActionRunapp;c:\win95\calc.exe")

' ssActionPlaysound example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
' Clicking myobj during show plays sound file applause.wav one time.
' Sound is embedded in PRZ file.
myobj.SetSSAction("ssActionPlaysound;c:\lotus\flg\media\
applause.wav;1;ssActionMediaEmbed")

' ssActionPlaymovie example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
' Clicking myobj during show plays the movie in file shark.aim continuously,
'  in center of screen, at fast speed.  Movie is embedded in PRZ.
myobj.SetSSAction("ssActionPlaymovie;c:\lotus\flg\media\
shark.aim;0;ssActionMediaEmbed;ssActionMediaLocationCenter;ssActionMediaPlayspeedFast"
)

' ssActionRunshow example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionRunshow;c:\lotus\work\flg\nextshow.prz")

' ssActionJumptourl example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionJumptourl;http://www.lotus.com")

' ssActionNoaction example
Dim myobj as DrawObject
Set myobj = CurrentPage.CreateRect
myobj.SetSSAction("ssActionNoaction")
```

Freelance Graphics: SetViewMode method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SETVIEWMODE_METHOD_EXSCRIPT',1)} See example

Set the view mode.

Syntax

documentobject.**SetViewMode(***mode*, *pagenumber***)**

Parameters

mode as Variant

Value	Description
$ViewDraw	Draw view
$ViewOutliner	Outliner view
$ViewSorter	Page Sorter view
$ViewSlideShow	Screen Show view

pagenumber as Integer

Page number to view.

Return values

None

```
' Example: SetViewMode method
' Sets the view mode to outliner and puts the cursor on page 1.
CurrentDocument.SetViewMode $ViewOutliner, 1
```

Freelance Graphics: Show method
{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes
{button ,AL(`H_FLG_SHOW_METHOD_EXSCRIPT',1)} See example
Makes the specified document the active document.

Syntax
documentobject.**Show**

Parameters
None

Return values
None

Usage
If you are working with multiple documents, use this command to switch between them.

```
' Example: Show method
' Make two new documents, name them, then use the Show property
' to make first one and then the other document the active document,
' also print the name of the active document in each case.
Set doc1 = CurrentApplication.NewDocument

doc1.SaveAs "Logo1"
Set doc2 = CurrentApplication.NewDocument
doc2.SaveAs "PRImag"
doc1.Show
Print CurrentDocument.Name
doc2.Show
Print CurrentDocument.Name
```

Freelance Graphics: StartGuidedTemplate method
{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_STARTGUIDEDTEMPLATE_METHOD_EXSCRIPT',1)} See example

Start using a content topic.

Syntax
documentobject.**StartGuidedTemplate**(*TemplateName*)

Parameters
*templatename a*s *String*

Return values
None

```
' Example: StartGuidedTemplate method
' Starts using a guided template in the current document.
CurrentDocument.StartGuidedTemplate "busrev.smc"
```

```
' Example: StopGuidedTemplate method
' Stops using a guided template in the current document.
CurrentDocument.StopGuidedTemplate
```

Freelance Graphics: StopGuidedTemplate method

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_STOPGUIDEDTEMPLATE_METHOD_EXSCRIPT',1)} See example

Stop using a content topic.

Syntax

documentobject.**StopGuidedTemplate**

Parameters

None

Return values

None

Freelance Graphics: StopPlay method

{button ,AL(`H_FLG_MEDIA_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_STOPPLAY_METHOD_EXSCRIPT',1)} See example

Stop playing the media.

Syntax

mediaobject.**StopPlay**

Parameters

None

Return values

None

```
' Example: StopPlay method
' Adds a movie to the page, plays it, and stops it.
Dim Movie As DrawObject
Set Movie = CurrentPage.CreateMovie("c:\lotus\flg\media\dinosaur.aim")
Movie.Media.Play
Movie.Media.StopPlay
```

```
' Example: Stretch method
' Creates a rectangle and stretches it.
Dim Rectangle As DrawObject
Set Rectangle = CurrentPage.CreateRect (1000, 2000, 3000, 3000)
Rectangle.Stretch 1000, 2000, 4000, 6000, 4000, 6000, 0
```

Freelance Graphics: Stretch method

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_STRETCH_METHOD_EXSCRIPT',1)} See example

Stretch the object.

Syntax

*drawobject***.Stretch(***xanchor***,** *yanchor***,** *xstart***,** *ystart***,** *xfinish***,** *yfinish***,** *stretchmode***)**

Parameters

All x and y specifications are twips.

xanchor as Long

Horizontal coordinate of the point from which the object will be stretched.

yanchor as Long

Vertical coordinate of the point from which the object will be stretched.

xstart as Long

Horizontal starting coordinate of the stretch vector.

ystart as Long

Vertical starting coordinate of the stretch vector.

xfinish as Long

Horizontal ending coordinate of the stretch vector.

yfinish as Long

Vertical ending coordinate of the stretch vector.

stretchmode as Integer

Must be zero.

Return values

None

Freelance Graphics: Tile method

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TILE_METHOD_EXSCRIPT',1)} See example

Tiles all document windows within the Freelance Graphics application window.

Syntax

windowobject.**Tile**

Parameters

None

Return values

None

```
' Example: Tile method
CurrentApplicationWindow.Tile
```

```
' Example: Ungroup method
Dim MySel As Selection
Set MySel = Selection
MySel.Group
MySel.Move 50,50
MySel.Ungroup
```

Freelance Graphics: Ungroup method
{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_UNGROUP_METHOD_EXSCRIPT',1)} See example

Ungroup a grouped object.

Syntax
drawobject.**Ungroup**

Parameters
None

Return values
None

Freelance Graphics LotusScript Methods A-Z

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

L

LeaveEditMode method

M

Maximize method
Minimize method
Move method

N

NearestColorFromRGB method
NewDocument method

O

OpenDocument method
OpenDocumentFromInternet
OpenDocumentFromNotes
OpenPresForCopy method

P

Paste method
PastePage method
PasteSelectedPages method
PasteSpecial method
Play method
Print method
PrintOut method
PublishToWeb
PutIntoPlacementBlock method

Q

Quit method

R

Remove method
RemoveFromSelection method
Replicate method
Restore method
RevertToStyle method
RGBToColor method
Rotate method
RunDialog method

S

SameColor method
Save method
SaveAs method
SaveAsToInternet
SaveAsToNotes
Select method
SelectPageForCopy method
SetInternetOptions
SetObjectData method
SetSSAction method
SetViewMode method
Show method
StartGuidedTemplate method
StopGuidedTemplate method
StopPlay method
Stretch method

Freelance Graphics LotusScript Properties A-Z

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Count property
CurrentPrinter property

D

DefaultFilePath property
Delay property
Description property
DimPrevious property
DisplayCoords property
DisplayDrawRuler property
DisplayGrid property
DisplayTextRuler property
DocName property
Document property
Documents property
DocWindow property
DoubleUnderline property

E

Embedded property
Exclude property
ExeName property

F

FirstIndent property
Font property
FontColor property
FontName property
FontUnit property
FullName property

G

Green property

H

Height property
HorizontalAlignment property

I

ID property
Interactive property
IsChart property
IsGroup property
IsImage property
IsMedia property
IsOleObj property
IsOpen property
IsPlacementBlock property
IsSelectable property
IsTable property
IsText property
IsValid property
Italic property

J, K

(None)

L

Layout property

Sound property
SpeakerNoteText property
StartNumber property
StartupView property
StrikeThrough property
Style property
SubScript property
SuperScript property

T
Table property
TemplateDir property
TemplatePageCount property
Text property
TextBlock property
TextProperties property
TextTightness property
Title property
Top property
TransitionEffect property
Type property

U
Underline property
UndoEnabled property
UnitOfMeasure property
UserClassNameApplication
UserClassNameFull
UserClassNameShort

V
VersionID property
VerticalAlignment property
ViewMode property
Visible property

W
WaitForClick property
Width property
WordDoubleUnderline property
WordUnderline property
WorkDir property

X, Y, Z
(None)

Freelance Graphics: ActiveDocument property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ACTIVEDOCUMENT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the active presentation document.

Data type

Document

Syntax

set *documentobject* **=** *applicationobject*.**ActiveDocument**

Legal values

Any instance of the Document class.

```
' Example: ActiveDocument property
' Makes Freelance the active document.
Dim DocObject As Document
Set DocObject = CurrentApplication.ActiveDocument
```

```
' Example: ActiveDocWindow property
' Makes Freelance the active window.
Dim DocObject As DocWindow
Set DocObject = CurrentApplication.ActiveDocWindow
```

Freelance Graphics: ActiveDocWindow property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ACTIVEDOCWINDOW_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the document window of the active presentation.

Data type

DocWindow

Syntax

set *documentwindow* **=** *applicationobject***.ActiveDocWindow**

Legal values

Any instance of the DocWindow class.

```
' Example: ActivePage property
' This example uses a variable to hold the active or current page.
' If you move to another page, making it the active page, you
' can return to the page you were on by refering to the named
' variable, called PageObj in this example.
' For this example to work you must have a presentation with at least
' three pages, and the initial "active page" should be a page other than
' page three.
Dim PageObj As Page
' Set the active page to the page object
Set PageObj = CurrentDocument.ActivePage
' Go to another page
CurrentDocument.GoToPage(3)
' Make the previously active page the active page again.
Set CurrentDocument.ActivePage = pageObj
```

Freelance Graphics: ActivePage property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ACTIVEPAGE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the current page, or if in Page Sorter view, the selected page.

Data type

Page

Syntax

set *page* **=** *documentobject***.ActivePage**

set *documentobject***.ActivePage =** *page*

Legal values

Any instance of the Page class.

```
' Example: Active property
' This property exists for compatibility reasons.
' It will always return True.
' Returns true in the IsActive variable
Dim DocObject As Document
Dim IsActive As Integer
Set DocObject = CurrentDocument
IsActive = DocObject.Active
```

Freelance Graphics: Active property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ACTIVE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the document is the active presentation.

Data type

Integer (Boolean)

Syntax

value **=** *documentobject***.Active**

Legal values

Value	Description
TRUE (non-0)	Document is the active presentation
FALSE (0)	Document is not the active presentation

Freelance Graphics: ApplicationWindow property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_APPLICATIONWINDOW_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the Freelance Graphics application window.

Data type

ApplicationWindow

Syntax

set *applicationwindow* **=** *applicationobject***.ApplicationWindow**

Legal values

Any instance of the ApplicationWindow class.

```
' Example: ApplicationWindow property
' Makes the current application the current window.
Dim AppWin As ApplicationWindow
Set AppWin = CurrentApplication.ApplicationWindow
```

Freelance Graphics: Application property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_DOCUMENT_CLASS;H_FLG_BASEOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_APPLICATION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the application.

Data type

Application

Document

String

Syntax

set *object* **=** *objecttype***.Application**

Legal values

Any instance of the Application or Document class; for the BaseObject class, returns "CurrentApplication."

```
' Example: Application property
' Sets the application object the current application
Dim AppObj As Application
Set AppObj = CurrentDocument.Application
' Note you could also have Set AppObj = CurrentApplicaton
' and achieved the same result.
' Example: Author property
' This script prints the author's name then assigns another name as author.
Dim AName As String
AName = CurrentDocument.Author
' Print the current author's name to the output window
Print AName
AName = "Leslie Smith"
CurrentDocument.Author = AName
' Print the new author's name to the output window
Print AName
```

Freelance Graphics: Author property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_AUTHOR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the author of the presentation.

Data type

String

Syntax

authorname **=** *documentobject***.Author**

*documentobject***.Author =** *authorname*

Legal values

Any string value.

```
' Example: AutoSaveInterval property
' Automatically save every five minutes.
CurrentApplication.Preferences.AutoSaveInterval = 5
```

Freelance Graphics: AutoSaveInterval property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_AUTOSAVEINTERVAL_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the number of minutes between automatic saves.

Note Used only if AutoSave property is TRUE (non-0).

Data type

Integer

Syntax

autosaveinterval **=** *preferencesobject*.**AutoSaveInterval**

preferencesobject.**AutoSaveInterval =** *autosaveinterval*

Legal values

Any integer.

Freelance Graphics: AutoSave property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_AUTOSAVE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Automatic save preference.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject*.**Autosave**

preferencesobject.**Autosave=** *flag*

Legal values

Value	Description
TRUE (non-0)	Perform automatic saves
FALSE (0)	Do not perform automatic saves

Usage

The time interval for automatic saves is stored in the AutoSaveInterval property.

```
' Example: AutoSave property
' Turn on autosave.
CurrentApplication.Preferences.AutoSave = True
```

Freelance Graphics: AutoTime property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_AUTOTIME_PROPERTY_EXSCRIPT',1)} See example

(Read-write) During a screen show, controls whether the page displays for the time interval specified in the *pageobject*.Delay property.

Data type

Integer (Boolean)

Syntax

value **=** *pageobject***.AutoTime**

*pageobject***.AutoTime =** *value*

Legal values

Value	Description
TRUE (non-0)	Page displays for the default time interval specified in *pageobject*.Delay
FALSE (0)	Page remains displayed until the user selects another page

```
' Example: AutoTime property
' Finds out whether Freelance Graphics will display page for the
' time interval.  Also sets the AutoTime property to true so that
' the display time interval specified will be use.
Dim TimeVal As Integer
TimeVal = CurrentPage.AutoTime
' Print whether the page will display for a specific time interval
Print TimeVal
TimeVal = 1
' Set AutoTime to 1 (True).
CurrentPage.AutoTime = TimeVal
'Print the result in the output window.
Print CurrentPage.AutoTime
```

```
' Example: BackColor property
' Make the background color of 2nd rectangle the same as the 1st.
Dim MyColor As Color
Set MyColor = MyRect1.Background.BackColor
Set MyRect2.Background.BackColor = MyColor
```

Freelance Graphics: BackColor property

{button ,AL(`H_FLG_BACKGROUND_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BACKCOLOR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the background color of a drawn object.

Data type

Color

Syntax

set *color* **=** *backgroundobject***.BackColor**

set *backgroundobject***.BackColor =** *color*

Legal values

Any instance of the Color class.

Freelance Graphics: Background property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BACKGROUND_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Background property of the object.

Data type

Background

Syntax

set *background* **=** *drawobject***.Background**

set *drawobject***.Background =** *background*

Legal values

Any instance of the Background class.

Usage

You can set a Background to an existing Background, or you can set the individual properties (BackColor, Color, and Pattern) of the Background class.

```
' Example: Background property
' Make the oval background the same as the rectangle background.
Dim MyBackgroundStyle As Background
Set MyBackgroundStyle = MyRect.Background
Set MyOval.Background = MyBackgroundStyle
```

```
' Example: BackupDir property
' Change the backup directory to c:\backup.
CurrentApplication.Preferences.BackUpDir = "c:\backup"
```

Freelance Graphics: BackupDir property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BACKUPDIR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the backup directory preference.

Data type

String

Syntax

backupdirectory **=** *preferencesobject***.BackupDir**

*preferencesobject***.BackupDir =** *backupdirectory*

Legal values

Any directory.

```
' Example: BlackWhitePal property
' Change the palette to black and white.
CurrentApplication.Preferences.BlackWhitePal = True
```

Freelance Graphics: BlackWhitePal property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BLACKWHITEPAL_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Black-and-white or Color palette flag.

Print using the gray scale equivalent of colors.

This property is the equivalent of checking the "Disable black & white palettes" option in the Freelance Graphics Preferences dialog box. To see this dialog box, choose File - User Setup - Freelance Preferences.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject*.**BlackWhitePal**

preferencesobject.**BlackWhitePal =** *flag*

Legal values

Value	**Description**
TRUE (non-0)	Enable black-and-white palette
FALSE (0)	Disable black-and-white palette

```
' Example: Blue property
' Find the amount of blue in an object's RGB color.
Dim AmountOfBlue As Integer
AmountOfBlue = MyRect.Background.Color.Blue
```

Freelance Graphics: Blue property

{button ,AL(`H_FLG_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BLUE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the amount of blue in a Color object.

Data type

Integer

Syntax

blueamount **=** *colorobject*.**Blue**

Legal values

0 (no blue) to 255 (maximum blue).

```
' Example: Bold property
' Make text bold.
MyText.TextBlock.Font.Bold = True
```

Freelance Graphics: Bold property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BOLD_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine if the font is bold.

Data type

Integer (Boolean)

Syntax

value **=** *fontobject***.Bold**

*fontobject***.Bold =** *value*

Legal values

Value	Description
TRUE (non-0)	Bold
FALSE (0)	Not bold

```
' Example: BorderDisplay property
' Show the printable area border.
CurrentApplication.Preferences.BorderDisplay = $BorderDispPrintableArea
```

Freelance Graphics: BorderDisplay property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BORDERDISPLAY_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the border display preference.

Data type

Integer (Variant)

Syntax

borderdisplay **=** *preferencesobject*.**BorderDisplay**

preferencesobject.**BorderDisplay =** *borderdisplay*

Legal values

Value	Description
$BorderDispMargin	Display drawing area border (recommended)
$BorderDispPrintableArea	Display printable area border
$BorderDispNone	Display no border

```
' Example: Border property
' Make the border of one rectangle 2 the same as rectangle 1, then
' change the border of rectangle 1 to thin.
' Give rectangle 2 the same border as rectangle 1.
Set MyRect2.Border = MyRect1.Border
' Give rectangle 1 a thin border line.
MyRect1.Border.Width = $ltsBorderWidthThin
```

Freelance Graphics: Border property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BORDER_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Border style for a drawn object.

Data type

Border

Syntax

set *border* **=** *drawobject***.Border**

set *drawobject***.Border =** *border*

Legal values

Any instance of the Border class.

Usage

You can set a Border to an existing Border, but more commonly you will set the individual properties (Color, Pattern, and Width) of the Border class.

```
' Example: BuildBullets property
' Make the created text block into a bullet build during a screen show.
Dim Txt As DrawObject
' Create textblock
Set Txt = CurrentPage.CreateText
' Set the bullet build feature on for this text block.
Txt.BuildBullets = 1
```

Freelance Graphics: BuildBullets property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BUILDBULLETS_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determines if a text block is to be made into a bullet build during a screen show.

Data type

Integer (Boolean)

Syntax

value **=** *drawobject*.**Buildbullets**

drawobject.**BuildBullets =** *value*

Legal values

Value	Description
TRUE (non-0)	Make text block into a bullet build during a screen show
FALSE (0)	Do not make text block into a bullet build during a screen show

```
' Example: BulletProperties property
' Creates a textblock, enters text, gets the bullet properties, and
' prints the size of the text to the output window.
Dim Txt As DrawObject
Dim Bullets As BulletProperties
Set Txt = CurrentPage.CreateText
Txt.Textblock.Text = "Some text"
Set Bullets = txt.BulletProperties
Print Bullets.Size
```

Freelance Graphics: BulletProperties property

{button ,AL(`H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_BULLETPROPERTIES_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the bullet properties for a text block.

Data type

BulletProperties

Syntax

set *bulletproperties* **=** *textblockobject*.**BulletProperties**

Legal values

Any instance of the BulletProperties class.

Usage

You can access the individual properties (Color, ShadowColor, ShadowDepth, and so on) of the BulletProperties class.

Freelance Graphics: Case property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CASE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the font case (upper or lower).

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer

Syntax

fontcase **=** *fontobject***.Case**

*fontobject***.Case=** *fontcase*

Legal values

Any integer (always returns zero).

```
' Example: Case property
' This property is only for compatibility reasons, it has no function
' in Freelance Graphics.
' Example: Changed property
' Determines whether the current document has changed, prints
' this value to the output window, then sets the Changed property
' to 1 (True).
Dim IsChanged As Integer
' Determined whethere the document has changed.
IsChanged = CurrentDocument.Changed
Print IsChanged
IsChanged = 1
' Set the Changed property to 1 (True), i.e. that it has changed.
CurrentDocument.Changed = IsChanged  ' Sets the changed property to 1
Print CurrentDocument.Changed
```

Freelance Graphics: Changed property
{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CHANGED_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine if the document has changed.

Data type
Integer (Boolean)

Syntax
value **=** *documentobject***.Changed**

*documentobject***.Changed =** *value*

Legal values

Value	Description
TRUE (non-0)	Document changed
FALSE (0)	Document has not been changed

```
' Example: Chart property
' This example shows how to get access to chart properties.
'  It first creates a chart then enters values, then makes it a three-dimensional
chart.
Dim ChartObj As DrawObject
Dim MyChart As Variant
Set ChartObj = CurrentPage.CreateChart
ChartObj.Series(1).DataPoints(1).Value = 25
ChartObj.Series(1).DataPoints(2).Value = 43
Set MyChart = ChartObj.Chart
MyChart.is3d = True
```

Freelance Graphics: Chart property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CHART_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the Chart property for a drawn object.

Note The Freelance Graphics Chart class is derived from the DrawObject class, and has all the properties and methods of the LotusChart ChartBase class. For more information about the LotusChart ChartBase class, see the Help contents under LotusScript, LotusChart LotusScript Reference, By Category, Classes.

Data type

Chart

Syntax

set *chart* **=** *drawobject***.Chart**

Legal values

Any instance of the Chart class.

Freelance Graphics: ColCount property

{button ,AL(`H_FLG_TABLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COLCOUNT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the number of columns in a table.

Data type

Integer

Syntax

columns **=** *tableobject***.ColCount**

Legal values

Any positive integer.

```
' Example: ColCount property
' Creates a table and prints the number of columns it has.
Dim TableObj As DrawObject
Set TableObj = CurrentPage.CreateTable (1, 4, 4)
Print TableObj.ColCount
```

```
' Example: Colors property
' Three ways to set the variable MyPeachColor to
' a peach color.
Dim MyPeachColor As Color
Set MyPeachColor = CurrentApplication.Colors.Item(178)
' or, next example:
Set MyPeachColor = CurrentApplication.Colors.Item("peach")
' or, next example:
Set MyPeachColor = CurrentApplication.Colors.RGBToColor(16764301)
```

Freelance Graphics: Colors property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COLORS_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the color palette for an application.

Data type

Colors

Syntax

set *colorpalette* **=** *colorsobject***.Colors**

Legal values

Any instance of the Colors class.

Usage

You can use this property to retrieve palette colors using the RGBToColor, Item, and GetNearestColor methods.

```
' Example: Color property
' This example sets the color of the rectangle to the
' same color as the line.
Dim MyColor As Color
Dim MyLine As DrawObject
Dim MyRect As DrawObjectSet MyLine = CreateLine
Set MyRect = CreateRect
Set MyColor = MyLine.LineStyle.Color
Set MyRect.Background.Color = MyColor
```

Freelance Graphics: Color property

{button ,AL(`H_FLG_BACKGROUND_CLASS;H_FLG_BORDER_CLASS;H_FLG_BULLET_PROPERTIES_CLASS;H_FLG_LINESTYLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COLOR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the color for a background, border, bullet, or line style.

Data type

Color

Syntax

set *color* **=** *object***.Color**

set *object***.Color =** *color*

Legal values

Any instance of the Color class.

Usage

You can set a Color to an existing Color, or you can use the RGBtoColor method to create a new Color from a combination of red, green, and blue. However, once you define a Color object in Freelance Graphics, you cannot modify its Red, Green, or Blue properties individually.

Freelance Graphics: Count property

{button ,AL(`H_FLG_COLORS_CLASS;H_FLG_DOCUMENTS_CLASS;H_FLG_OBJECTS_CLASS;H_FLG_PAGES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_COUNT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the number of open presentations, the number of objects in a collection, the number of colors in the palette, or the number of pages in a collection.

Data type
Integer

Syntax
count **=** *object*.**Count**

Legal values
Any integer.

```
' Example: Count property
' Two examples of how to use the count property.
' The first prints the number of pages in the presentation.
' The second prints the number of objects on the current page.
' Note that the underscore is used as a line continuation
' character in LotusScript.
Print "There are " + str$(CurrentDocument.Pages.Count) + _
   " pages in this document."
' or, next example:
Dim NumObjs As Integer
NumObjs = CurrentPage.Objects.Count
Print "There are " + str$(NumObjs) + " objects on this page."
```

Freelance Graphics: CurrentPrinter property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_CURRENTPRINTER_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the name of the current printer for an application.

Data type

String

Syntax

printername **=** *applicationobject***.CurrentPrinter**

Legal values

Any printer name.

```
' Example: CurrentPrinter property
' Get the name of the current printer for Freelance Graphics
' and print the name in the output window.
Dim PrinterName As String
PrinterName = CurrentApplication.CurrentPrinter
Print PrinterName
```

```
' Example: DefaultFilePath property
' Print the default file pate of Freelance Graphics.
Print "The default file path is " + CurrentApplication.DefaultFilePath
```

Freelance Graphics: DefaultFilePath property
{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DEFAULTFILEPATH_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the application's default file path.

Data type
String

Syntax
defaultfilepath **=** *applicationobject*.**DefaultFilePath**

Legal values
Any file path.

Freelance Graphics: Delay property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DELAY_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the time delay (in seconds) for a drawn object or page transition during a screen show. For a page, this delay is used only when *pageobject*.AutoTime is TRUE (non-0).

Data type

Integer

Syntax

secondsdelay **=** *object***.Delay**

object **.Delay =** *secondsdelay*

Legal values

Any positive integer.

```
' Example: Delay property
' Set the screen show time delay for a rectangle on the page.
Dim Rec As DrawObject
Set Rec = CurrentPage.CreateRect
Dim Del As Integer
Del = Rec.Delay     ' Get the delay setting
If Del < 3 then
   Del = 3
   Rec.Delay = Del  ' Set the delay setting to 3 seconds.
End If
```

```
' Example: Description property
' Get and print the existing description for a document, then
' change the document description.
Dim Desc As String
' Gets the existing description for the document.
Desc = CurrentDocument.Description
' Print description to output window.
' Change the description of the document.
Desc = "This document has been modified by script."
CurrentDocument.Description = Desc
' Print new description to output window.
Print CurrentDocument.Description
```

Freelance Graphics: Description property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_BACKGROUND_CLASS;H_FLG_BASEOBJECT_CLASS;H_FLG_CHART_CLASS;H_FLG_COLOR_CLASS;H_FLG_COLORS_CLASS;H_FLG_DOCUMENT_CLASS;H_FLG_DOCUMENTS_CLASS;H_FLG_DOCWINDOW_CLASS;H_FLG_DRAWOBJECT_CLASS;H_FLG_FONT_CLASS;H_FLG_OBJECTS_CLASS;H_FLG_OLEOBJECT_CLASS;H_FLG_PAGES_CLASS;H_FLG_PAGESELECTION_CLASS;H_FLG_PLACEMENTBLOCK_CLASS;H_FLG_SELECTION_CLASS;H_FLG_TABLE_CLASS;H_FLG_TEXTBLOCK_CLASS;H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DESCRIPTION_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the description of the presentation stored in the document, or get the class name of an object.

Note For chart objects this is a read only property.

Data type
String

Syntax
documentobject.**Description** = *description*

description **=** *documentobject*.**Description**

Legal values
Any string value.

Freelance Graphics: DimPrevious property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DIMPREVIOUS_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determines if the previous items in a bulleted list are to be dimmed during a bullet build (during a screen show).

Data type

Integer (Boolean)

Syntax

value **=** *drawobject***.DimPrevious**

*drawobject***.DimPrevious =** *value*

Legal values

Value	Description
TRUE (non-0)	Dim the previous bullets
FALSE (0)	Do not dim the previous bullets

```
' Example: DimPrevious property
' Create text, see if DimPrevious is true or false, if false, set to true.
Dim txt As DrawObject
Set txt = CurrentPage.CreateText
Dim value As Integer
' Get existing setting for this text block.
value = txt.DimPrevious
If (value = FALSE) then
   value = TRUE
   ' Set the dimPrevious setting to something new.
   txt.DimPrevious = value
End If
```

Freelance Graphics: DisplayCoords property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DISPLAYCOORDS_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Display coordinates preference.

Data type

Integer (Boolean)

Syntax

displaycoordinates **=** *preferencesobject*.**DisplayCoords**

preferencesobject.**DisplayCoords =** *displaycoordinates*

Legal values

Value	Description
TRUE (non-0)	Display the coordinates
FALSE (0)	Do not display the coordinates

```
' Example: DisplayCoords property
' Display the coordinates.
CurrentApplication.Preferences.DisplayCoords = True
```

Freelance Graphics: DisplayDrawRuler property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DISPLAYDRAWRULER_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Display drawing ruler preference.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject***.DisplayDrawRuler**

*preferencesobject***.DisplayDrawRuler =** *flag*

Legal values

Value	Description
TRUE (non-0)	Display the drawing ruler
FALSE (0)	Do not display the drawing ruler

```
' Example: DisplayDrawRuler property
' Display the draw ruler.
CurrentApplication.Preferences.DisplayDrawRuler = True
```

```
' Example: DisplayGrid property
' Display the grid.
CurrentApplication.Preferences.DisplayGrid = True
```

Freelance Graphics: DisplayGrid property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DISPLAYGRID_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Display grid preference.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject***.DisplayGrid**

*preferencesobject***.DisplayGrid =** *flag*

Legal values

Value	Description
TRUE (non-0)	Display grid
FALSE (0)	Do not display grid

```
' Example: DisplayTextRuler property
' Display the text ruler.
CurrentApplication.Preferences.DisplayTextRuler = True
```

Freelance Graphics: DisplayTextRuler property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DISPLAYTEXTRULER_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Display text ruler preference.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject***.DisplayTextRuler**

*preferencesobject***.DisplayTextRuler =** *flag*

Legal values

Value	Description
TRUE (non-0)	Display text ruler
FALSE (0)	Do not display text ruler

Freelance Graphics: DocName property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DOCNAME_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the document (file) name for the presentation.

Data type

String

Syntax

documentname **=** *documentobject***.DocName**

Legal values

Any string value.

```
' Example: DocName property
'Print the name of the document.
Dim DName As String
DName = CurrentDocument.DocName
Print DName
```

Freelance Graphics: Documents property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DOCUMENTS_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the collection of open presentations.

Data type

Documents

Syntax

set *documentcollection* **=** *applicationobject*.**Documents**

Legal values

Any instance of the Documents class (a collection of Document objects).

```
' Example: Documents property
' Two examples of using the Documents property:
' The first prints the names of all the documents that are currently open
' in Freelance Graphics.
' The second prints the number of documents that are currently open in
' Freelance Graphics.
ForAll doc in CurrentApplication.Documents
   Print doc.Name
End ForAll
' or, next example:
Dim num As Integer
num = CurrentApplication.Documents.Count
Print "There are " + str$(num) + " presentations open."
```

Freelance Graphics: Document property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DOCUMENT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the document containing the page.

Data type
Document

Syntax
set *documentobject* **=** *pageobject*.**Document**

Legal values
Any instance of the Document class.

Usage
You can set a Document object to an existing Document, or you can set the individual properties (Author, Description, Location, and so on) of the Document class.

```
' Example: Document property
' Sets the document property to DocObj.
Dim DocObj As Document
Set DocObj = CurrentPage.Document
```

```
' Example: DocWindow property
' Sets the document window property to DWindow.
Dim DWindow As DocWindow
Set DWindow = CurrentDocument.DocWindow
```

Freelance Graphics: DocWindow property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DOCWINDOW_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the document window size (width and height).

Data type

DocWindow

Syntax

set *documentwindow* **=** *documentobject***.DocWindow**

Legal values

Any instance of the DocWindow class.

Usage

Use this property to get the width and height of the document window.

Freelance Graphics: DoubleUnderline property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_DOUBLEUNDERLINE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the Double underline (two lines under both words and spaces) font property.

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer (Boolean)

Syntax

flag **=** *fontobject*.**DoubleUnderline**

fontobject.**DoubleUnderline =** *flag*

Legal values

Value	Description
TRUE (non-0)	Double underline
FALSE (0)	No double underline

```
' Example: DoubleUnderline property
' This property is provided for compatibility reasons only
```

Freelance Graphics: Embedded property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_EMBEDDED_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is embedded.

Data type

Integer (Boolean)

Syntax

flag **=** *documentobject*.**Embedded**

Legal values

Value	Description
TRUE (non-0)	Embedded object
FALSE (0)	Not an embedded object

```
' Example: Embedded property
' IsEmbedded will contain either a 0 or 1. If the current document were
' imbedded in a Notes document for example, IsEmbedded would be 1(True).
Dim IsEmbeded As Integer
IsEmbedded = CurrentDocument.Embedded
```

Freelance Graphics: Exclude property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_EXCLUDE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine if the page is excluded from the screen show.

Data type

Integer (Boolean)

Syntax

value **=** *pageobject***.Exclude**

*pageobject***.Exclude =** *value*

Legal values

Value	Description
TRUE (non-0)	Excluded from screen show
FALSE (0)	Not excluded from screen show

```
' Example: Exclude property
' See if the current page is excluded, if it is not, exclude it.
Dim value As Integer
' Determine whether the current page is excluded.
value = CurrentPage.Exclude
If (value = 0) Then
   ' Set value to CurrentPage.Exclude to specifically exclude a page.
   CurrentPage.Exclude = value
End If
```

Freelance Graphics: ExeName property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_EXENAME_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the executable to launch from an object during a screen show.

Data type

String

Syntax

executablename **=** *drawobject***.ExeName**

*drawobject***.ExeName =** *executablename*

Legal values

Any legal executable name.

```
' Example: ExeName property
' Examples of getting and setting the ExeName property.
Dim ProductName As String
' Get the ExeName of the product.
ProductName = BmpObj.ExeName
' Set the ExeName to launch the product with this draw object.
BmpObj.ExeName = ProductName
```

Freelance Graphics: FirstIndent property

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FIRSTINDENT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the First indent property for the text (the number of twips the first line of text is to be indented).

Data type

Integer

Syntax

indentation **=** *textobject*.**FirstIndent**

textobject.**FirstIndent =** *indentation*

Legal values

Any integer.

Usage

Use the ParaIndent property to set the amount of space you want each paragraph indented.

```
' Example: FirstIndent property
' Example of how to get and set the FirstIndent property.
Dim txt As DrawObject
Dim Indent As Integer
Set txt = CurrentPage.CreateText
txt.Text = "Some text"
' Get the existing first line indent.
Indent = txt.TextProperties.FirstIndent
' Set the first line indent.
txt.TextProperties.FirstIndent = Indent
```

```
' Example: FontColor property
' This example requires that two text objects be defined on the page:
' MyTextObject1 and MyTextObject2.  The font colors of each are set
' to the same color.
' Note that the underscore is used as a line continuation
' character in LotusScript.
Dim MyColor As Color
Set MyColor = CurrentApplication.Colors.Item(178)
Set MyTextObject1.TextBlock.Font.FontColor = MyColor

Set MyTextObject2.TextBlock.Font.FontColor = _
      CurrentApplication.Colors.Item(178)
```

Freelance Graphics: FontColor property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FONTCOLOR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the color for a font.

Data type

Color

Syntax

set *color* **=** *fontobject***.FontColor**

set *fontobject***.Color =** *color*

Legal values

Any instance of the Color class.

Usage

You can set a FontColor to an existing Color, or you can use the Colors class to set a FontColor to a Color from the Freelance Graphics palette.

```
' Example: FontName property
' This example requires that you have an object on the page defined
' as MyTextObject. The font for the text is set to Arial.
MyTextObject.TextBlock.Font.FontName = "Arial"
```

Freelance Graphics: FontName property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FONTNAME_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine or set   the Font name for a Font class.

Data type

String

Syntax

fontname **=** *fontobject***.FontName**

*fontobject***.FontName =** *fontname*

Legal values

Any valid font name.

Freelance Graphics: FontUnit property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FONTUNIT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine the font units of measurement for an application. For Freelance Graphics, this is always points.

Data type

Variant

Syntax

fontunit **=** *applicationobject***.FontUnit**

Legal values

The only legal value is $ltsScaleModePoint.

```
' Example: FontUnit property
' Get the Freelance Graphics font unit.
Unit = CurrentApplication.FontUnit
```

Freelance Graphics: Font property

{button ,AL(`H_FLG_TEXTBLOCK_CLASS;H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FONT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the font property for the TextProperties or TextBlock class.

Data type

Font

Syntax

set *font* **=** *object*.**Font**

set *object*.**Font =** *font*

Legal values

Any instance of the Font class.

Usage

You can set a Font to an existing Font, or you can set the individual properties (Bold, Case, FontColor, and so on) of the Font class.

```
' Example: Font property
' Make the font bold.
Dim MyFont As Font
Dim MyTextObject As DrawObject
Set MyTextObject = CurrentPage.CreateText

MyTextObject.TextBlock.Text = "This is an example."
Set MyFont = MyTextObject.TextBlock.Font
MyFont.Bold = True
```

Freelance Graphics: FullName property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_FULLNAME_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the full application or document path name (the full path name includes the executable file name).

Data type

String

Syntax

fullname **=** *object***.FullName**

Legal values

Any valid full path name.

```
' Example: FullName property
' Provides the full name of the current document.
DocName = CurrentDocument.FullName
```

```
' Example: Green property
' Find out the amount of green in an object.
' You must have a rectangle named MyRect on the page.
Dim AmountOfGreen As Integer
AmountOfGreen = MyRect.Background.Color.Green
```

Freelance Graphics: Green property

{button ,AL(`H_FLG_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_GREEN_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the amount of green in a Color object.

Data type

Integer

Syntax

greenamount **=** *colorobject***.Green**

*colorobject***.Green=** *greenamount*

Legal values

0 (no green) to 255 (maximum green).

```
' Example: Height property
' Gets the height of the rectangle object created with default values.
Dim H As Long
Dim RectObj As DrawObject
Set RectObj = CurrentPage.CreateRect
H = RectObj.Height
```

Freelance Graphics: Height property

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCWINDOW_CLASS;H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_HEIGHT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the application window, document window, or drawn object height, in twips.

Data type

Integer (AppWindow, DocWindow classes)

Long (DrawObject class)

Syntax

twipshigh **=** *object***.Height**

Legal values

Any positive value.

```
' Example: HorizontalAlignment property
' Setting and getting the HorizontalAlignment property.
Dim txt As DrawObject
Dim Horiz As Variant
Set txt = CurrentPage.CreateText
txt.Text = "Some text"
' Get the horizontal alignment of the text.
Horiz = txt.TextProperties.HorizontalAlignment
Horiz = $ltsAlignmentHorizCenter
' Set the new horizontal alignment.
txt.TextProperties.HorizontalAlignment = Horiz
```

Freelance Graphics: HorizontalAlignment property

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_HORIZONTALALIGNMENT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the horizontal alignment property for the text.

Data type

Variant (Enumerated)

Syntax

horizontalalignment **=** *textobject*.**HorizontalAlignment**

HorizontalAlignment = *horizontalalignment*

Legal values

Value	Description
$ltsAlignmentLeft	Left aligned
$ltsAlignmentRight	Right aligned
$ltsAlignmentHorizCenter	Centered horizontally
$ltsAlignmentJustify	Fit between margins

```
' Example: ID property
'Finds and prints the id's of the placement blocks on the page.
Dim id As Integer
Forall obj In CurrentPage.Objects
   If (obj.IsPlacementBlock) Then
      id = obj.ID
     Print id
   End If
End Forall
```

Freelance Graphics: ID property

{button ,AL(`H_FLG_PLACEMENTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ID_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the ID for a "Click here..." block.

Data type

Integer

Syntax

idnumber **=** *placementblockobject***.ID**

Legal values

Any "Click here..." block number.

```
' Example: Interactive property
' Find out if the application is interactive.
Dim value As Integer
value = CurrentApplication.Interactive
```

Freelance Graphics: Interactive property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_INTERACTIVE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the application is interactive. For Freelance Graphics, this value is always TRUE (non-0).

Data type

Integer (Boolean)

Syntax

value **=** *applicationobject*.**Interactive**

Legal values

Value	Description
TRUE (non-0)	Interactive
FALSE (0)	Not interactive

Freelance Graphics: IsChart property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISCHART_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is a chart object.

Data type

Integer (Boolean)

Syntax

value **=** *drawobject***.IsChart**

Legal values

Value	Description
TRUE (non-0)	Chart
FALSE (0)	Not a chart

```
' Example: IsChart property
' Finds all chart objects on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsChart) Then
      Print "Chart"
   End If
End Forall
```

```
' Example: IsGroup property
' Determine if the selected object is a grouped object.
' If it is, print the text and then ungroup the object.
' There must be a selected grouped object on the page for this
' script to work.
If Selection.IsGroup Then
   Print "Selected DrawObject is a Grouped object."
   Selection.UnGroup
End If
```

Freelance Graphics: IsGroup property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISGROUP_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is a grouped object.

Data type

Integer (Boolean)

Syntax

value **=** *drawobject***.IsGroup**

Legal values

Value	Description
TRUE (non-0)	Grouped
FALSE (0)	Not grouped

Freelance Graphics: IsImage property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISIMAGE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is an image object (a bitmap or graphics metafile, for example).

Data type

Integer (Boolean)

Syntax

value **=** *drawobject***.IsImage**

Legal values

Value	Description
TRUE (non-0)	Image
FALSE (0)	Not an image

```
' Example: IsImage property
' Finds all images on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsImage) Then
      Print "Image"
   End If
End Forall
```

```
' Example: IsMedia property
' Finds all media objects on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsMedia) Then
      Print "Media"
   End If
End Forall
```

Freelance Graphics: IsMedia property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISMEDIA_PROPERTY_EXSCRIPT',1)} See example

(Read-only) TRUE if object is a media object (a movie or sound, for example).

Data type

Integer (Boolean)

Syntax

value **=** *drawobject*.**IsMedia**

Legal values

Value	Description
TRUE (non-0)	Movie or sound object
FALSE (0)	Not a movie or sound object

```
' Example: IsOleObj property
' Finds all OLE objects on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsOLEObj) Then
      Print "OLE"
   End If
End Forall
```

Freelance Graphics: IsOleObj property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISOLEOBJ_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is an OLE object.

Data type

Integer (Boolean)

Syntax

value **=** *drawobject***.IsOleObj**

Legal values

Value	Description
TRUE (non-0)	OLE object
FALSE (0)	Not an OLE object

Freelance Graphics: IsOpen property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISOPEN_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the presentation document is currently open. In Freelance Graphics, this is always TRUE (non-0).

Data type

Integer (Boolean)

Syntax

value **=** *documentobject***.IsOpen**

Legal values

Value	Description
TRUE (non-0)	Open
FALSE (0)	Not open

```
' Example: IsOpen property
' Tells whether the document is open;
' this is always True in Freelance Graphics
If (CurrentDocument.IsOpen) Then
   Print "Open"
Else
   Print "Not open"
End If
```

```
' Example: IsPlacementBlock property
' Finds all placement blocks on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsPlacementBlock) Then
      Print "PlacementBlock"
   End If
End Forall
```

Freelance Graphics: IsPlacementBlock property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISPLACEMENTBLOCK_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is a "Click here..." block.

Data type

Integer (Boolean)

Syntax

value **=** *drawobject*.**IsPlacementBlock**

Legal values

Value	Description
TRUE (non-0)	"Click here..." block
FALSE (0)	Not a "Click here..." block

Freelance Graphics: IsSelectable property

{button ,AL(`H_FLG_DOCUMENT_CLASS;H_FLG_DRAWOBJECT_CLASS;H_FLG_BASEOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISSELECTABLE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object can be selected.

Data type
Integer (Boolean)

Syntax
value **=** *object***.IsSelectable**

Legal values

Value	**Description**
TRUE (non-0)	Selectable (objects of the DrawObject and Page classes)
FALSE (0)	Not selectable (all other classes)

```
' Example: IsSelectable property
' Finds all selectable objects on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsSelectable) Then
      Print "Selectable"
   End If
End Forall
```

```
' Example: IsTable property
' Finds all table objects on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsTable) Then
      Print "Table"
   End If
End Forall
```

Freelance Graphics: IsTable property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISTABLE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is a table object.

Data type

Integer (Boolean)

Syntax

value **=** *drawobject***.IsTable**

Legal values

Value	Description
TRUE (non-0)	Table
FALSE (0)	Not a table

Freelance Graphics: IsText property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ISTEXT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is a text object

Data type

Integer (Boolean)

Syntax

value **=** *drawobject***.IsText**

Legal values

Value	Description
TRUE (non-0)	Text object
FALSE (0)	Not a text object

```
' Example: IsText property
' Finds all text objects on the page.
Forall obj In CurrentPage.Objects
   If (obj.IsText) Then
      Print "Text"
   End If
End Forall
```

Freelance Graphics: IsValid property

{button ,AL(`;H_FLG_DOCUMENT_CLASS;H_FLG_DRAWOBJECT_CLASS;H_FLG_BASEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_ISVALID_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the object is valid (still available).

Data type

Integer (Boolean)

Syntax

value **=** *object***.IsValid**

Legal values

Value	Description
TRUE (non-0)	Valid
FALSE (0)	Not valid

```
' Example: IsValid property
' The following example prints the value zero (FALSE).
' You must have a rectangle on the page identified as MyRect.
MyRect.Cut
Print MyRect.IsValid
```

```
' Example: Italic property
' Make text italic.
Dim MyFont As Font
Dim MyTextObject As DrawObject
Set MyTextObject = CurrentPage.CreateText
MyTextObject.TextBlock.Text = "This is an example."
MyTextObject.TextBlock.Font.Italic = True
```

Freelance Graphics: Italic property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ITALIC_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the italic attribute for the font.

Data type

Integer (Boolean)

Syntax

value **=** *font***.Italic**

*font***.Italic =** *value*

Legal values

Value	Description
TRUE (non-0)	Italicized
FALSE (0)	Not italicized

Freelance Graphics: Layout property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_LAYOUT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the layout used by the page.

Data type

String

Syntax

layoutname **=** *pageobject***.Layout**

*pageobject***.Layout =** *layoutname*

Legal values

Any existing page layout name. Use a zero-length string ("") to refer to the [Blank Page] layout.

Tip Use the Freelance Graphics user interface to display all page layout names.

```
' Example: Layout property
' Find the layout being used by the page, set page to the
' Bulleted List property.
Dim L As String
L = CurrentPage.Layout
L = "Bulleted List"
CurrentPage.Layout = L
```

Freelance Graphics: Left property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_LEFT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the left edge of the object, in twips.

Data type

Long

Syntax

lefttwip **=** *drawobject***.Left**

Legal values

Any positive value.

```
' Example: Left property
' Gets the left edge of the rectangle.
Dim L As Long
Dim Rec As DrawObject
Set Rec = CurrentPage.CreateRect
L = Rec.Left
```

```
' Example: LineLead property
' Setting and getting the LineLead property.
Dim Leading As Long
Dim txt As DrawObject
Set txt = CurrentPage.CreateText
' Get the leading text value.
Leading = txt.TextProperties.LineLead
' Set the leading text value.
txt.TextProperties.LineLead = Leading
```

Freelance Graphics: LineLead property

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_LINELEAD_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the line leading for the text (the number of points between line skips).

Data type

Integer

Syntax

lineleading **=** *textobject***.LineLead**

*textobject***.LineLead=** *lineleading*

Legal values

Any positive integer.

```
' Example: LineStyle property
' Create a line, then make its width thin.
Dim Myline1 As DrawObject
Set Myline1 = CurrentPage.CreateLine(1000,1000,4000,4000)
MyLine1.LineStyle.Width = $ltsBorderWidthThin
```

Freelance Graphics: LineStyle property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_LINESTYLE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the line style property.

Data type

LineStyle

Syntax

set *linestyleobject* **=** *drawobject***.LineStyle**

set *drawobject***.LineStyle =** *linestyleobject*

Legal values

You can set a LineStyle to an existing LineStyle, or you can set the individual properties (Color, Pattern, and Width) of the LineStyle class.

```
' Example: Location property
' Print the location of the document.
Dim Path As String
Path = CurrentDocument.Location
Print Path
```

Freelance Graphics: Location property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_LOCATION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the application or document path.

Data type

String

Syntax

path **=** *object***.Location**

Legal values

Any valid path name.

Freelance Graphics: Media property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_MEDIA_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the Media class properties for an object. This property is valid only if *drawobject*.isMedia is TRUE (non-0).

Data type

Media

Syntax

set *media* **=** *drawobject***.Media**

Legal values

Any instance of the Media class.

```
' Example: Media property
' Create and play a media object.
Dim MovieObj As Media
Dim MedObject As DrawObject
Set MedObject = CurrentPage.CreateMovie("c:\lotus\flg\media\aircraft.aim")
' Get the media properties.
Set MovieObj = MedObject.Media
MovieObj.play
```

Freelance Graphics: Name property

{button ,AL(`H_FLG_COLOR_CLASS;H_FLG_DRAWOBJECT_CLASS;H_FLG_PAGE_CLASS;H_FLG_DOCUMENT_CLASS;H_FLG_BASEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_NAME_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the page or drawn object name; for any other type of Freelance Graphics object this property can only get the name.

Note Read-only except for the Page and DrawObject classes.

Data type

String

Syntax

name **=** *object*.**Name**

object.**Name** = *name*

Legal values

Any ASCII string value, but may not contain a semi-colon (;) or an equal sign (=).

```
' Example: Name property
' Create a rectangle, then name the rectangle.
' Note that when an object is created it is selected.
' Name the current page. Finally, print the names
' of the rectangle and the page.
Dim MyRect As DrawObject
CurrentPage.CreateRect
Selection.Name = "purple rectangle"
Set MyRect = CurrentPage.FindObject("purple rectangle")
CurrentPage.Name = "My Title Page"
Print MyRect.Name
Print CurrentPage.Name
```

```
' Example: Number property
' Get and print the current page number in the Output window.
Dim pagenum As Integer
' Get the current page number
pagenum = CurrentPage.Number
Print pagenum
```

Freelance Graphics: Number property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_NUMBER_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the page number.

Data type

Integer

Syntax

pagenumber **=** *pageobject***.Number**

Legal values

From 1 to the number of pages in the presentation.

Usage

To move a page, set the page number to the new page number.

Freelance Graphics: Objects property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_OBJECTS_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get a collection of drawing objects on a page.

Data type

ObjectCollection

Syntax

set *objectcollection* **=** *pageobject*.**Objects**

Legal values

Any collection of objects.

```
' Example: Objects property
' Cycle through all the objects on the page and print their names.
' Then find the number of objects on a page and print the number.
ForAll objs in CurrentPage.Objects
   Print objs.name
End ForAll
Dim MyPage As Page
Dim num As Integer
Set MyPage = CurrentPage
num = MyPage.Objects.Count
Print "There are " + Str$(num) + " objects on this page."
```

Freelance Graphics: ObjectType property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_OBJECTTYPE_PROPERTY_EXSCRIPT',1)} See example

Read only property that indicates the Freelance internal object type for DrawObjects.

Data type

Variant

Syntax

typename = *object*.**ObjectType**

Legal values

DrawObject class only.

Value	Description
DrawObjectTypeNone	There are no draw objects on the page.
DrawObjectTypeList	List--may only show up with older Freelance Graphics file formats.
drawObjectTypeRectangle	Rectangle
DrawObjectTypeLine	Line
DrawObjectTypeText	Text
DrawObjectTypePolyline	Polyline
DrawObjectTypePolygon	Polygon
DrawObjectTypePie	Pie
DrawObjectTypeWincontrol	Window control
DrawObjectTypeGroup	Group
DrawObjectTypeClientobject	Client object
DrawObjectTypeSeries	Series--may only show up with older Freelance Graphics file formats.
DrawObjectTypeSymbol	Clip art or diagram
DrawObjectTypeShape	Shape
DrawObjectTypeStext	Curved text
DrawObjectTypeEllipse	Ellipse
DrawObjectTypeArc	Arc
DrawObjectTypeImage	Image
DrawObjectTypePolyrect	Polyrect--may only show up with older Freelance Graphics file formats.
DrawObjectTypePolytic	Polytic--may only show up with older Freelance Graphics file formats.
DrawObjectTypePolytext	Polytext--may only show up with older Freelance Graphics file formats.
DrawObjectTypeTable	Table
DrawObjectTypeMetafile	Metafile
DrawObjectTypeMetafiletext	Metafile text
DrawObjectTypeAfid	Afid

DrawObjectTypeBackdrop	Backdrop
DrawObjectTypeOrgchart	Orgchart
DrawObjectTypeChart	Chart
DrawObjectTypeOle	Ole object
DrawObjectTypePb	Placement block
DrawObjectTypeConnector	Connector
DrawObjectTypeTextshape	Textshape
DrawObjectTypeMedia	Media
DrawObjectTypePbg	Placement block guide
DrawObjectTypeComment	Comment

```
' Example: ObjectType property
Sub printtypes
  Forall DrawObj In CurrentPage.Objects
     Print DrawObj.ObjectType
  End Forall
End Sub
```

Freelance Graphics: Object property

{button ,AL(`H_FLG_OLEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_OBJECT_PROPERTY_EXSCRIPT',1)} See example

(Read only) Returns the embedded OLE object's automation interface.

Data type

Variant.

Syntax

set *MyOLE* = *OLEObject*.**Object**

Usage

This property returns the embedded object's native automation interface, so that you can access its methods, properties, and so on.

```
' Example: Object property
' This example finds the identified Lotus Draw Component
'(an OLE object) on the page, then
' if there is an existing drawing it deletes the drawing, it
' then adds a drawing from the file
' graph.dgm
' There must be a Lotus Draw Component as an OLE object on
' the current page for this example to work. The OLE object can
' be added through the user interface or through LotusScript,
' see CreateObject method to find out how to
' add an OLE object using LotusScript in Freelance Graphics.
Dim DrawOCX As DrawObject
Dim OCXAuto As Variant
' When an OLE object is on the page,
' Freelance Graphics assigns a name to the object;
' in this example, the name is Lotus Draw/Diagram Component1.
' Brackets are shorthand notation in LotusScript
' for the object with the specified name on the current
' page. In this script the object variable, DrawOCX,
' is set to the Lotus Draw Component on the
' current page.
Set DrawOCX = [Lotus Draw/Diagram Component1]
' Get the OCX's automation interface.
Set OCXAuto = DrawOCX.Object
' Then use interface to access the OCX's scripting language.
' Note, you must know the OCX's scripting language.
' Select and clear all drawings from the Draw/Diagram Component.
OCXAuto.SelectAll
OCXAuto.Selection.Clear
' Add a new drawing; in this case the fourth diagram in graph.dgm.
OCXAuto.CreateSymbol "C:\lotus\Smasters\FLG\graph.dgm" , 4
```

Freelance Graphics: OffsetReplicate property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_OFFSETREPLICATE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Offset replicate preference.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject*.**OffsetReplicate**

preferencesobject.**OffsetReplicate =** *flag*

Legal values

Value	Description
TRUE (non-0)	Offset copy from original
FALSE (0)	Place copy on top of original

```
' Example: OffsetReplicate property
' Instruct Freelance Graphics to offset all future replicas.
CurrentApplication.Preferences.OffsetReplicate = True
```

```
' Example: OleObj property
' Use the variable Obj as a handle to manipulate an OLE object.
' However, you must know the API of the OLE object to atually
' manipulate the object.
Dim Obj As Variant
Dim Rect As DrawObject
Set Rect = CurrentPage.CreateObject ("Bitmap image")
' Get the properties of the OLE object.
Set Obj = Rect.OleObj
```

Freelance Graphics: OleObj property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;H_FLG_OLEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_OLEOBJ_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get an OLE object.

Data type

Variant

Syntax

set *object* **=** *drawobject***.OleObj**

Legal values

As defined by the application that created the object.

```
' Example: Overstrike property
' Provided for compatibility reasons; it is ignored by Freelance Graphics.
```

Freelance Graphics: Overstrike property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_OVERSTRIKE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the overstrike attribute for the font. The overstrike characters are typed over existing characters, usually indicating a deletion.

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer (Boolean)

Syntax

value **=** *FontRef*.**Overstrike**

FontRef.**Overstrike =** *value*

Legal values

Value	Description
TRUE (non-0)	Overstrike on
FALSE (0)	Overstrike off

Freelance Graphics: PageSelection property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PAGESELECTION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the set of pages currently selected in an application or document.

Data type

PageSelection

Syntax

set *pageselection* **=** *object***.PageSelection**

Legal values

Any page selection.

```
' Example: PageSelection property
' Gets the page selection from the current document.
Dim sel As PageSelection
Set sel = Currentdocument.PageSelection
```

```
' Example: Pages property
' Print the names of all the pages in the presentation.
' Print the number of pages in the presentation.
ForAll page in CurrentDocument.Pages
   Print page.name
End ForAll
Dim num As Integer
Dim MyDocument As Document
Set MyDocument = CurrentDocument
num = MyDocument.Pages.Count
Print "There are " + str$(num) + " pages in MyDocument."
```

Freelance Graphics: Pages property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PAGES_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get pages in the presentation document.

Data type

PageCollection

Syntax

set *pagecollection* **=** *documentobject***.Pages**

Legal values

Any collection of pages in the document.

```
' Example: PageTransitionDelay property
' Find the transition delay value, print out the value, then
' change the transition delay.
Dim Del As Integer
' Get the current transition delay.
Del = CurrentDocument.PageTransitionDelay
' Print the value of Del.
Print Del
' Set the delay value to another integer.
Del = 5
' Set the transition delay property to the new value.
CurrentDocument.PageTransitionDelay = Del
```

Freelance Graphics: PageTransitionDelay property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PAGETRANSITIONDELAY_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the default delay for each page in the presentation when running a screen show.

Data type

Integer

Syntax

secondsdelay **=** *documentobject***.PageTransitionDelay**

*documentobject***.PageTransitionDelay =** *secondsdelay*

Legal values

Any positive integer.

```
' Example: PageTransitionEffect property
' Find out the current transition effect value, print it, then
' change it to a new value.
Dim Effect As Variant
' Get the current transition effect value
Effect = CurrentDocument.PageTransitionEffect
' Print the effect value.
Print Effect
' Set Effect to a new value.
Effect = 5
' Apply the new value to the transition effect property
CurrentDocument.PageTransitionEffect = Effect
```

Freelance Graphics: PageTransitionEffect property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PAGETRANSITIONEFFECT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the screen show page transition effect for each page in the presentation.

Data type

Variant

Syntax

pagetransitioneffect **=** *documentobject***.PageTransitionEffect**

*documentobject***.PageTransitionEffect =** *pagetransitioneffect*

Legal values

Value	Description
$SSPageReplace	Instant replacement
$SSPageBottom	From bottom
$SSPageLeft	From left
$SSPageRight	From right
$SSPageBlinds	Blinds opening
$SSPageLouvers	Louvers opening
$SSPageBlocks	Block replacement
$SSPageCenter	From center out
$SSPageBoxIn	From outer box in
$SSPageZigZag	Zigzag replacement
$SSPageHorzln	Horizontal lines
$SSPageHVertln	Vertical lines
$SSPageTop	From top
$SSPageBoxOut	From inner box out
$SSPageHorizOut	Slide out horizontally
$SSPageVertOut	Slide out vertically
$SSPageFade	Fade to new
$SSPageDiagL	Diagonal left
$SSPageDiagR	Diagonal right
$SSPagePanL	Pan left
$SSPagePanR	Pan right
$SSPageScrollT	Scroll from the top
$SSPageScrollB	Scroll from the bottom
$SSPageDraw	Draw new page
$SSPageRain	Rain new page
$SSPagePBrush	Paintbrush new page
$SSPageShade	Shade new page
$SSPageCurtain	Open curtain
$SSPageBMPCol	Bitmap by colors

```
' Example: Page property
' Creates a table on the current page and prints the page number
' where the table is located.
Dim Pg As Page
Dim Table As DrawObject
Set Table = CurrentPage.CreateTable (1, 4, 4)
Set Pg = Table.Page
Print Pg.Number
```

Freelance Graphics: Page property
{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes
{button ,AL(`H_FLG_PAGE_PROPERTY_EXSCRIPT',1)} See example
(Read-only) Get the page on which the drawn object exists.

Data type
Page

Syntax
set *pageobject* **=** *drawobject***.Page**

Legal values
Any instance of the Page class.

```
' Example: ParaIndent property
' Get the paragraph indent setting, then reset this setting to 2000.
Dim Txt As DrawObject
Dim Indent As Long
Set Txt = CurrentPage.CreateText
Txt.Text = "Some text"
' Get the current paragraph indent setting.
Indent = Txt.Textproperties.ParaIndent
' Set the paragraph indent to a new value.
Indent = 2000
' Apply the new value to the paragraph indent property.
Txt.TextProperties.ParaIndent = Indent
```

Freelance Graphics: ParaIndent property

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PARAINDENT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Paragraph indent property for the text (the number of twips to indent the paragraph).

Data type

Integer

Syntax

indentation **=** *textobject***.ParaIndent**

*textobject***.ParaIndent =** *indentation*

Legal values

Any integer.

Usage

Use the FirstIndent property to set the amount of space you want first line of the paragraph indented.

```
' Example: Paralead property
' Find out what the value of the Paralead property is, print it out,
' then change it.
Dim value As Long
Dim txt As DrawObject
Set txt = CurrentPage.CreateText
txt.Text = "Some Text"
' Get the current value of the paralead property.
value = txt.TextProperties.Paralead
' Print the paralead value.
Print value
' Set the value to 2 1/2 spacing.
value = 150
' Assign the new value to the paralead property.
txt.TextProperties.Paralead = value
```

Freelance Graphics: Paralead property

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS',0)} See list of classes

{button ,AL(`H_FLG_PARALEAD_PROPERTY_EXSCRIPT',1)} See example

(Read-Write) Defines the spacing between paragraphs in a text block as a percentage of a single-spaced line.

Data type

Integer

Syntax

value = *textpropertyobject*.**Paralead**

textpropertyobject.**Paralead** = value

Legal values

Percentage of a single-spaced line. For example, 100 indicates double spacing, 0 indicates single spacing, and 50 indicates single spacing plus a half a space, that is, a space and a half. Legal values are from 0 to 900.

Usage

This property does in script what the user interface provides in the Paragraph value of the "Space between Lines, Paragraphs" area of the InfoBox when a text block is selected (but not being edited). The values for the parameter are related to those shown in the InfoBox by the formula: Infoboxvalue = (Paralead + 100) /   100.

Freelance Graphics: Parent property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_DOCUMENT_CLASS;H_FLG_BASEOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PARENT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Identify the parent application from which the current application or document was launched; or return the string "CurrentApplication" if the object is not a member of the Application or Document class.

Data type

Application

Document

Syntax

set *parent* **=** *object***.Parent**

Legal values

Any instance of the Application or Document class, or the string "CurrentApplication."

```
' Example: Parent property
' Identifies the parent application containing the current document.
Dim ParentApp As Application
Set ParentApp = CurrentDocument.Parent
```

Freelance Graphics: Path property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PATH_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the path of the application or document.

Data type

String

Syntax

path **=** *applicationobject***.Path**

Legal values

Any legal path.

```
' Example: Path property
' Gets the path of the current application and prints it.
Dim DocPath As String
DocPath = CurrentApplication.Path
Print DocPath
```

Freelance Graphics: Pattern property

{button ,AL(`H_FLG_BACKGROUND_CLASS;H_FLG_BORDER_CLASS;H_FLG_LINESTYLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PATTERN_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set a background, border, or line style pattern.

Data type

Variant

Syntax

pattern = *object*.**Pattern**

object.**Pattern** = *pattern*

Legal values

There are separate values for each class. You can view the patterns in the InfoBox of the Freelance Graphics user interface for the appropriate object type, and match them with the names below.

Border class

Value	Description
$ltsBorderPatternNone	No border
$ltsBorderPatternSolid	Solid border
$ltsBorderPatternDashDot	Dash-dot border
$ltsBorderPatternDashDotDot	Dash-dot-dot border
$ltsBorderPatternLongDash	Long-dash border
$ltsBorderPatternDashed	same as $ltsBorderPatternDot
$ltsBorderPatternDot	same as $ltsBorderPatternDashed

LineStyle class

Value	Description
$ltsLineStyleNone	No line
$ltsLineStyleSolid	Solid line
$ltsLineStyleDashDot	Dash-dot line
$ltsLineStyleDashDotDot	Dash-dot-dot line
$ltsLineStyleLongDash	Long-dash line
$ltsLineStyleMediumDash	same as $ltsLineStyleDot
$ltsLineStyleDot	same as $ltsLineStyleMediumDash

Background values

Value	Description
$ltsFillNone	No fill
$ltsFillSolid	100% foreground color
$ltsFillGray1	90% foreground color
$ltsFillGray2	80% foreground color
$ltsFillGray3	70% foreground color
$ltsFillGray4	60% foreground color
$ltsFillGray5	50% foreground color
$ltsFillGray6	40% foreground color
$ltsFillGray7	30% foreground color
$ltsFillGray8	20% foreground color
$ltsFillGray9	10% foreground color

$ltsFillGray10	same as $ltsFillGray9
$ltsFillLeftDiagonal	Left diagonal fill lines
$ltsFillRightDiagonal	Right diagonal fill lines
$ltsFillDiagonalHatch	Diagonal hatch fill lines
$ltsFillHorizontal	Horizontal fill lines
$ltsFillVertical	Vertical fill lines
$ltsFillRegularHatch	Regular hatch fill lines
$ltsFillLeftRightGrad	Left-right gradation
$ltsFillBottomTopGrad	Bottom-top gradation
$ltsFillNeToSwGrad	Northeast to southwest gradation
$ltsFillNwToSeGrad	Northwest to southeast gradation
$ltsFillCenterBoxGrad	Center box gradation
$ltsFillLowBoxGrad	Lower box gradation
$ltsFillCenterCircleGrad	Center circle gradation
$ltsFillLowCircleGrad	Lower circle gradation
$ltsFillNeToSwDiagonalStripGrad	Northeast to southwest diagonal gradation
$ltsFillNwToSeDiagonalStripGrad	Northwest to southeast diagonal gradation

```
' Example: Pattern property
' Make the rectangle a dashed outline and a gray fill background.
' Make the line dash-dot style.
Dim MyRect As DrawObject
Dim MyLine As DrawObject
Set MyRect = CurrentPage.CreateRect
Set MyLine = CurrentPage.CreateLine

MyRect.Border.Pattern = $ltsBorderPatternDashed
MyRect.Background.Pattern = $ltsFillGray8
MyLine.LineStyle.Pattern = $ltsLineStyleDashDot
```

Freelance Graphics: PlacementBlock property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PLACEMENTBLOCK_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the "Click here..." block properties for a drawn object. These properties are valid only when *drawobject*.isPlacementBlock is TRUE (non-0).

Data type

PlacementBlock

Syntax

set *placementblock* **=** *placementblockobject***.PlacementBlock**

Legal values

Any instance of the PlacementBlock class.

```
' Example: PlacementBlock property
' Find a placement block and assign a variable name to it.
Dim PbProp As PlacementBlock
Forall obj In CurrentPage.Objects
   If (obj.IsPlacementBlock) Then
      Set PbProp = obj.PlacementBlock
   End If
End Forall
```

```
' Example: PlayPriority property
' Create a rectangle and find out its play priority, print the
' value of the play priority, then change the priority.
Dim Play As Integer
Dim Rect As DrawObject
Set Rect = CurrentPage.CreateRect
' Get the current Play priority of the rectangle.
Play = Rect.PlayPriority
' Set a new value for the play priority.
Play = 2
' Assign the new value to the play priority property.
Rect.PlayPriority = Play
```

Freelance Graphics: PlayPriority property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PLAYPRIORITY_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the priority with which the object transitions onto the page during a screen show.

Data type

Integer

Syntax

playpriority **=** *drawobject***.PlayPriority**

*drawobject***.PlayPriority =** *playpriority*

Legal values

1=first object, 2=second, and so on

```
' Example: Preferences property
' Turn autosave on.
CurrentApplication.Preferences.AutoSave = True
```

Freelance Graphics: Preferences property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PREFERENCES_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the preferences for the application.

Data type

Preferences

Syntax

set *preferences* **=** *applicationobject*.**Preferences**

set *applicationobject*.**Preferences.***individualproperty = preference*

Legal values

Any instance of the Preferences class.

Usage

You can get the set of preferences in a Preferences object, or you can get or set the individual properties (AutoSave, AutoSaveInterval, BackupDir, and so on) of the Preferences class.

Freelance Graphics: Priority property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PRIORITY_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the drawing priority of a drawn object.

Note You can set the priority of a DrawObject that is a placement block only if you are editing a .MAS or .SMC file, otherwise it is read-only.

Data type

Integer

Syntax

priority **=** *drawobject***.Priority**

*drawobject***.Priority =** *priority*

Legal values

1 = first, 2 = second, and so on

Usage

If there are empty placement blocks on the page, they are always lowest in priority (that is, a priority of one), so that an empty placement block is always behind another drawn object. If you try to put something other than an empty placement block at a priority equal to or lower than the placement block's priority, you get a message explaining that it cannot be done.

Note The priority of a DrawObject of any kind can never be set higher than the total number of DrawObjects on the page. Also, a DrawObject, other than a PlacementBlock, cannot be lower in priority than a PlacementBlock.

```
' Example: Priority property
' Create a rectangle and find out its priority, print the
' value of the priority, then change the priority.
Dim Play As Integer
Dim Rect As DrawObject
Set Rect = CurrentPage.CreateRect
' Get the current  priority of the rectangle.
Play = Rect.Priority
Print Play
' Assign the new value to the priority property.
Play = 2
Rect.Priority = Play
```

Freelance Graphics: PromptText property

{button ,AL(`H_FLG_PLACEMENTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_PROMPTTEXT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the prompt text for a "Click here..." block.

Data type

String

Syntax

prompttext **=** *placementblockobject***.PromptText**

Legal values

Any string value.

```
' Example: PromptText property
Sub Main
Dim ptext As String
   Forall objs In CurrentPage.Objects
      If objs.isplacementblock Then
         ptext = objs.prompttext
      End If
      Print ptext
   End Forall
End Sub
```

```
' Example: ReadOnly property
' Finds out if the current document is read-only.
Dim IsReadOnly As Integer
IsReadOnly  = CurrentDocument.ReadOnly
```

Freelance Graphics: ReadOnly property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_READONLY_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine if the presentation is read-only.

Data type

Integer (Boolean)

Syntax

value **=** *documentobject***.ReadOnly**

Legal values

Value	Description
TRUE (non-0)	Read-only
FALSE (0)	Read-write

```
' Example: Red property
' Make the variable "AmountOfRed" equal to the red value (in RGB terms)
' of the background color of the rectangle and print the value.
' You must have a rectangle variable on the page named MyRect for this
' script to work.
Dim AmountOfRed As Integer
AmountOfRed = MyRect.Background.Color.Red
Print Str$(AmountOfRed)
```

Freelance Graphics: Red property

{button ,AL(`H_FLG_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_RED_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the amount of red in a Color object.

Data type

Integer

Syntax

redamount **=** *colorobject***.Red**

Legal values

0 (no red) to 255 (maximum red).

```
' Example: RightIndent property
' Create a text block, enter some text in it, print the value of
' the right indent property, then change the right indent value.
Dim Indent As Integer
Dim txt As DrawObject
Set txt = CurrentPage.CreateText
txt.Text = "Some text"
' Get the current right indent property and print it.
Indent = txt.TextProperties.RightIndent
Print Indent
' Change the indent value.
Indent = 300
' Set the right indent property to the new value.
txt.TextProperties.RightIndent = Indent
```

Freelance Graphics: RightIndent property

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_RIGHTINDENT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Right indent property for the text (the number of twips to indent the text).

Data type

Integer

Syntax

indentation **=** *textobject***.RightIndent**

*textobject***.RightIndent =** *indentation*

Legal values

Any integer.

Freelance Graphics: RowCount property

{button ,AL(`H_FLG_TABLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_ROWCOUNT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the number of rows in a table.

Data type

Integer

Syntax

rows **=** *tableobject***.RowCount**

Legal values

Any positive integer.

```
' Example: RowCount property
' Create a table, then print out the number of rows in it.
Dim rows As Integer
Dim Table As DrawObject
Set Table = CurrentPage.CreateTable (1, 4, 4)
rows = Table.RowCount
Print rows
```

Freelance Graphics: ScanSpeed property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SCANSPEED_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Scan speed preference (the number of seconds delay between displaying images when scanning in the browser).

Data type

Long

Syntax

secondsdelay **=** *preferencesobject***.ScanSpeed**

*preferencesobject***.ScanSpeed =** *secondsdelay*

Legal values

Any value between .1 and 100.0.

```
' Example: ScanSpeed property
' Specify a 3 second scan speed.
CurrentApplication.Preferences.ScanSpeed = 3.0
```

Freelance Graphics: SelectedObjects property

{button ,AL(`H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SELECTEDOBJECTS_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the collection of currently selected objects.

Data type

Objects

Syntax

set *selection* **=** *objectsobject***.SelectedObjects**

set *objectsobject***.SelectedObjects =** *selection*

Legal values

Any selection of objects.

Usage

This property is useful for saving and then restoring a selection.

```
' Example: SelectedObjects property
' Clears and restores selection.
Dim MySelection As Objects
Set MySelection = Selection.SelectedObjects
Selection.ClearSelection
Set Selection.SelectedObjects = MySelection
```

Freelance Graphics: SelectionCount property

{button ,AL(`H_FLG_PAGESELECTION_CLASS;H_FLG_SELECTION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SELECTIONCOUNT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the number of selected pages or objects.

Data type

Integer

Syntax

selectioncount **=** *object***.SelectionCount**

Legal values

Any integer.

```
' Example: SelectionCount property
' Print the number of selected objects on a page.
Dim num As Integer
num = Selection.SelectionCount
Print "There are " + str$(num) + " selected objects."
```

```
' Example: Selection property
' Group all of the selected objects on the page.
Dim MySelection As Selection
Set MySelection = CurrentPage.Selection
MySelection.Group
```

Freelance Graphics: Selection property

{button ,AL(`H_FLG_APPLICATION_CLASS;H_FLG_DOCUMENT_CLASS;H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SELECTION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get a selection of objects in the application or document, or on the currently active page.

Data type

Selection

Syntax

set *selection* **=** *object***.Selection**

Legal values

Any selection of objects.

Freelance Graphics: ShadowColor property

{button ,AL(`H_FLG_BULLET_PROPERTIES_CLASS;H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SHADOWCOLOR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Shadow color for a bullet or text.

Data type

Color

Syntax

set *color* **=** *shadowobject***.ShadowColor**

set *shadowobject***.ShadowColor =** *color*

Legal values

Any instance of the Color class.

```
' Example: ShadowColor property
' Create a text object, give it a shadow, get the shadow
' color of the text, then change the shadow color to another color.
Dim col As Color
Dim NewColor As Color
Set NewColor = CurrentApplication.Colors.RGBToColor(2998564)
Dim Txt As DrawObject
Set Txt = CurrentPage.CreateText
Txt.Text = "Some text"
' Give the text a shadow.
Set Txt.TextProperties.ShadowDirection = $ltsShadowBottomRight
' Get the current shadow color of the text block.
Set col = Txt.TextProperties.ShadowColor
'Set a new color to the shadow color property.
Set Txt.TextProperties.ShadowColor = NewColor
```

```
' Example: ShadowDepth property
' Create text, give it a shadow, get the current shadow depth, then
' make the shadow deeper.
Dim Depth As Variant
Dim Txt As DrawObject
Set Txt = CurrentPage.CreateText
Txt.Text = "Some text"
' Give the text a shadow.
Set Txt.TextProperties.ShadowDirection = $ltsShadowBottomRight
' Get the current shadow depth property.
Depth = Txt.TextProperties.ShadowDepth
' Set the depth to a new value.
Depth = $ltsShadowDepthDeep
' Apply the new depth to the shadow depth property.
Txt.TextProperties.ShadowDepth = Depth
```

Freelance Graphics: ShadowDepth property

{button ,AL(`H_FLG_BULLET_PROPERTIES_CLASS;H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SHADOWDEPTH_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the shadow depth for a bullet or text.

Data type

Variant (Enumerated)

Syntax

shadowdepth **=** *shadowobject*.**ShadowDepth**

shadowobject.**ShadowDepth =** *shadowdepth*

Legal values

Value	Description
$ltsShadowDepthShallow	Short shadow
$ltsShadowDepthNormal	Normal shadow
$ltsShadowDepthDeep	Long shadow

Freelance Graphics: ShadowDirection property

{button ,AL(`H_FLG_BULLET_PROPERTIES_CLASS;H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SHADOWDIRECTION_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Shadow direction for a bullet or text.

Data type

Variant (Enumerated)

Syntax

shadowdirection **=** *shadowobject***.ShadowDirection**

*shadowobject***.ShadowDirection =** *shadowdirection*

Legal values

Value	Description
$ltsShadowNone	No shadow
$ltsShadowBottomRight	Shadow on bottom right
$ltsShadowBottomLeft	Shadow on bottom left
$ltsShadowTopRight	Shadow on top right
$ltsShadowTopLeft	Shadow on top left

```
' Example: ShadowDirection property
' Create text, give it a shadow, get the shadow direction, then
' set the shadow to a new direction.
Dim Direction As Variant
Dim Txt As DrawObject
Set Txt = CurrentPage.CreateText
Txt.Text = "Some text"
' Give the text a shadow.
Set Txt.TextProperties.ShadowDirection = $ltsShadowBottomLeft
' Get the current shadow direction property
Direction = Txt.TextProperties.ShadowDirection
' Set the direction to a new value.
Direction = $ltsShadowBottomRight
' Apply the new direction to the property.
Txt.TextProperties.ShadowDirection = Direction
```

```
' Example: Size property
' Create a text block, get the point size, change the
' point size.
Dim txt As DrawObject
Dim pts As Double
Set txt = CurrentPage.CreateText
txt.TextBlock.Text = "Some text"
' Gets the point size for the text block.
pts = txt.TextBlock.Font.Size
' Change the point size.
pts = 25
' Sets the point size for the text block.
txt.TextBlock.Font.Size = pts
```

Freelance Graphics: Size property

{button ,AL(`H_FLG_BULLET_PROPERTIES_CLASS;H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SIZE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the font or bullet size (in points).

Data type

Double

Syntax

points **=** *object***.Size**

*object***.Size =** *points*

Legal values

Any positive integer.

Freelance Graphics: SkipWelcome property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SKIPWELCOME_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Skip welcome preference.

This property puts or removes a checkmark in the "Skip the startup dialogs..." checkbox in the Freelance Preferences dialog box. To see the Freelance Preferences dialog box, choose File - User Setup - Freelance Preferences.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject***.SkipWelcome**

*preferencesobject***.SkipWelcome =** *flag*

Legal values

Value	Description
TRUE (non-0)	Skip Welcome on entry
FALSE (0)	Display Welcome on entry

```
' Example: SkipWelcome property
' Do not display the Freelance Graphics welcome page.
CurrentApplication.Preferences.SkipWelcome = True
```

Freelance Graphics: SmallCaps property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SMALLCAPS_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine if the font displays as Small caps (smaller point size, but all uppercase).

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer (Boolean)

Syntax

value **=** *fontobject***.SmallCaps**

*fontobject***.SmallCaps =** *value*

Legal values

Value	Description
TRUE (non-0)	Small caps
FALSE (0)	No small caps

```
' Example: SmallCaps property
' Ignored by Freelance Graphics; it is here for compatibility reasons only.
' Example: SmartLook property
' Find out the SmartLook for the current document, print
' it out, then change the SmartLook to Brass.mas.
Dim Look As String
' Get the smartlook for the current document.
Look = CurrentDocument.SmartLook
' Print the SmartLook.
Print Look
' Change the SmartLook.
Look = "Brass.mas"
' Set the SmartLook for the current document.
CurrentDocument.SmartLook =Look
```

Freelance Graphics: SmartLook property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SMARTLOOK_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the SmartMaster look for the current presentation.

Data type

String

Syntax

smartlook **=** *documentobject*.**SmartLook**

documentobject.**SmartLook =** *smartlook*

Legal values

Any existing SmartMaster look.

Note You can use the LotusScript Dir command to list the *.MAS files in the \SMASTERS\FLG directory.

Freelance Graphics: SnapToGrid property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SNAPTOGRID_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Snap to grid preference.

Data type

Integer (Boolean)

Syntax

flag **=** *preferencesobject*.**SnapToGrid**

preferencesobject.**SnapToGrid =** *flag*

Legal values

Value	Description
TRUE (non-0)	Snap objects to grid
FALSE (0)	Do not snap objects to grid

```
' Example: SnapToGrid property
' Make objects align with the grid.
CurrentApplication.Preferences.SnapToGrid = True
```

```
' Example: Sound property
' Finds then changes the sound attached to a page.
Dim SoundName As String
' Gets the name of the sound attached to the page
SoundName = CurrentPage.Sound
SoundName = "applause.wav"
' Sets a new sound to the page
CurrentPage.Sound = SoundName
```

Freelance Graphics: Sound property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SOUND_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the file name for a sound associated with the page.

Data type

String

Syntax

soundfilename **=** *pageobject***.Sound**

*pageobject***.Sound =** *soundfilename*

Legal values

Any existing sound file name.

Freelance Graphics: SpeakerNoteText property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SPEAKERNOTETEXT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the speaker note text displayed on the page.

Data type

String

Syntax

speakernotetext **=** *pageobject***.SpeakerNoteText**

*pageobject***.SpeakerNoteText =** *speakernotetext*

Legal values

Any string value.

```
' Example: SpeakerNoteText property
' Get the existing speaker note, then change it.
Dim txt As String
' Get the existing speaker note text
txt = CurrentPage.SpeakerNoteText
' Change the speaker note text
txt = "Some more text"
' Set the speaker note text to something new
CurrentPage.SpeakerNoteText = txt
```

```
' Example: StartNumber property
' Create a text block, get the start number, then change it.
Dim num As Integer
Dim txt As DrawObject
Set txt = CurrentPage.CreateText
txt.TextBlock.Text = "Some text"
' Get the current bullet properties start number
num = txt.TextBlock.BulletProperties.StartNumber
' Set the bullet properties start number to something new.
num = 4
txt.TextBlock.BulletProperties.StartNumber = num
```

Freelance Graphics: StartNumber property

{button ,AL(`H_FLG_BULLETPROPERTIES_CLASS',0)} See list of classes

{button ,AL(`H_FLG_STARTNUMBER_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the starting number for numbered bullets.

Data type

Integer

Syntax

startnumber **=** *bulletobject***.StartNumber**

*bulletobject***.StartNumber =** *startnumber*

Legal values

Any integer.

Freelance Graphics: StartupView property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_STARTUPVIEW_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Startup view preference.

Data type

Variant (Enumerated)

Syntax

startupview **=** *preferencesobject*.**StartupView**

preferencesobject.**StartupView =** *startupview*

Legal values

Value	**Description**
$ViewDraw	Current Page view
$ViewOutliner	Outliner view
$ViewSorter	Page Sorter view
$ViewSlideShow	Screen Show view

```
' Example: StartUpView property
' Always open Freelance Graphics in the Sorter view.
CurrentApplication.Preferences.StartupView = $ViewSorter
```

```
' Example: StrikeThrough property
' Give the specified text the strikethrough attribute.
' For this example to work you must have a text object
' variable named MyTextObject on the page.
MyTextObject.TextBlock.Font.StrikeThrough = True
```

Freelance Graphics: StrikeThrough property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_STRIKETHROUGH_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine or set the strikethrough attribute for the font. Strikethrough is a horizontal line that prints through the middle of existing characters.

Data type

Integer (Boolean)

Syntax

value **= *fontobject*.StrikeThrough**

*fontobject***.StrikeThrough =** *value*

Legal values

Value	Description
TRUE (non-0)	Strikethrough on (~~for example~~)
FALSE (0)	Strikethrough off

```
' Example: Style property
' Create a text block, get the current bullet style, then
' change the bullet style.
Dim txt As DrawObject
Dim Style As Variant
Set txt = CurrentPage.CreateText
txt.TextBlock.Text = "Some text"
' Get the current bullet style.
Style = txt.TextBlock.BulletProperties.Style
' Set the bullet style to something different.
Style = $ltsBulletRomanNums
' Apply the new bullet style to the property.
txt.TextBlock.BulletProperties.Style = Style
```

Freelance Graphics: Style property

{button ,AL(`H_FLG_BULLETPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_STYLE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the Bullet style.

Data type

Variant (Enumerated)

Syntax

bulletstyle **=** *bulletpropertiesobject***.Style**

*bulletpropertiesobject***.Style =** *bulletstyle*

Legal values

Value	Description
$ltsBulletNone	None
$ltsBulletSmallLetters	
$ltsBulletCapLetters	
$ltsBulletRomanNums	
$ltsBulletDecimalNums	
$ltsBulletSmallDot	
$ltsBulletLargeDot	
$ltsBulletSmallSquare	
$ltsBulletLargeSquare	
$flwBulletDash	
$ltsBulletArrowhead	
$ltsBulletCheck	
$ltsBulletX	
$ltsBulletStar	
$ltsBulletPlus	
$flwBulletCurvedArrowhead	
$ltsbulletRndSquare	
$ltsBulletSmallDiamond	
$ltsBulletLargeDiamond	
$ltsBulletSmallArrowhead	

Freelance Graphics: SubScript property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SUBSCRIPT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the subscript attribute for the font. Subscripted characters are placed just below the surrounding text.

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer (Boolean)

Syntax

value **=** *fontobject*.**SubScript**

fontobject.**SubScript =** *value*

Legal values

Value	Description
TRUE (non-0)	Subscript on
FALSE (0)	Subscript off

```
' Example: SubScript property
' Ignored by Freelance Graphics; it is here for compatibility reasons only.
' Example: SuperScript property
' Ignored by Freelance Graphics; it is here for compatibility reasons only.
```

Freelance Graphics: SuperScript property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_SUPERSCRIPT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the superscript attribute for the font. Superscripted characters are placed just above the surrounding text.

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer (Boolean)

Syntax

value **= *fontobject*.SuperScript**

*fontobject***.SuperScript =** *value*

Legal values

Value	Description
TRUE (non-0)	Superscript on
FALSE (0)	Superscript off

```
' Example: Table property
' Create table, then get the table properties.
Dim TabProp As Table
Dim obj As DrawObject
Set obj = CurrentPage.CreateTable (1, 4, 4)
' Gets the table properties for the table
Set TabProp = obj.Table
```

Freelance Graphics: Table property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TABLE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get table properties for a drawn object. These properties are valid only when *drawobject*.IsTable is TRUE (non-0).

Data type

Table

Syntax

set *table* **=** *drawobject***.Table**

Legal values

Any instance of the Table class.

```
' Example: TemplateDir property
' Specify the content topic (SmartMaster) directory--in this
' example the directory is set to c:\lotus\smasters\flg.
CurrentApplication.Preferences.TemplateDir = "c:\lotus\smasters\flg"
```

Freelance Graphics: TemplateDir property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TEMPLATEDIR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the content topic (SmartMaster) directory preference.

Data type

String

Syntax

templatedirectory **=** *preferencesobject*.**TemplateDir**

preferencesobject.**TemplateDir =** *templatedirectory*

Legal values

Any directory.

```
' Example: TemplatePageCount property
' Gets the number of smartmaster content pages in the current document
Dim count As Integer
count = CurrentDocument.TemplatePageCount
```

Freelance Graphics: TemplatePageCount property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TEMPLATEPAGECOUNT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the content topic page count.

Data type

Integer

Syntax

pagecount **=** *documentobject***.TemplatePageCount**

Legal values

Any integer.

```
' Example: TextBlock property
' Create a text block, then get the text block properties.
Dim txt As Drawobject
Dim txtProps As TextBlock
Set txt = CurrentPage.CreateText
txt.TextBlock.Text = "Some text"
' Get the properties for the selected text block.
Set txtProps = txt.TextBlock
```

Freelance Graphics: TextBlock property
{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TEXTBLOCK_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the text block properties for a drawn object. These properties are valid only if *drawobject*.isText is TRUE (non-0).

Data type
TextBlock

Syntax
set *textblock* **=** *drawobject***.TextBlock**

Legal values
Any instance of the TextBlock class.

Freelance Graphics: TextProperties property

{button ,AL(`H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TEXTPROPERTIES_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the text properties for a text block.

Data type
TextProperties

Syntax
set *textproperties* **=** *textblockobject***.TextProperties**

Legal values
Any instance of the TextProperties class.

Usage
You can get a set of TextProperties, or you can get or set the individual properties (FirstIndent, Font, HorizontalAlignment, and so on) of the TextProperties class.

```
' Example: TextProperties property
' Create a text block, then get all of the properties for
' the text block.
Dim txt As DrawObject
Dim props As TextProperties
Set txt = CurrentPage.CreateText
txt.TextBlock.Text = "Some text"
' Get all of the text properties for the current text block
Set Props = txt.TextBlock.TextProperties
```

Freelance Graphics: TextTightness property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TEXTTIGHTNESS_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the Text tightness for the font.

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer

Syntax

tightness **=** *fontobject***.TextTightness**

*fontobject***.TextTightness =** *tightness*

Legal values

Any integer (always returns zero).

```
' Example: TextTightness property
' Ignored by Freelance Graphics; it is here for compatibility reasons only.
' Example: Text property
' Add text to the specified text object.
' For this example to work you must have a text object variable
' named MyTextObject on the page.
MyTextObject.TextBlock.Text = "This text will appear in the text block"
```

Freelance Graphics: Text property

{button ,AL(`H_FLG_TEXTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TEXT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the text contained in a text block.

Data type

String

Syntax

text **=** *textblockobject***.Text**

*textblockobject***.Text =** *text*

Legal values

Any string value.

Freelance Graphics: Title property

{button ,AL(`H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TITLE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the title text block on a page.

Data type

DrawObject

Syntax

set ***titleobject*** **=** ***pageobject*****.Title**

Legal values

Any instance of a page title text block.

Usage

You cannot set this property directly, however you can change the page title by changing the underlying Text object (*pageobject*.Title.Textblock.Text).

```
' Example: Title property
' Assign the title placement block to a variable, then add
' text to the placement block.
Dim TitleObj As DrawObject
' Get the title placement block and assign it to titleobj
Set TitleObj = CurrentPage.Title
' Set the text of the title block
TitleObj.Textblock.Text = "more text"
```

```
' Example: Top property
' Create Rectangle, find out what the edge is, then print out that value
' in the Output window.
Dim TopDimension As Long
Dim Rectangle As DrawObject
Set Rectangle = CurrentPage.CreateRect
' Get the top edge in twips
TopDimension = Rectangle.Top
Print TopDimension
```

Freelance Graphics: Top property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TOP_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the top edge of an object, in twips.

Data type

Long

Syntax

toptwip **=** *DrawObject***.Top**

Legal values

Any positive value.

Freelance Graphics: TransitionEffect property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;H_FLG_PAGE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TRANSITIONEFFECT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the transition effect for an object or page during a screen show.

Data type

Variant

Syntax

transitioneffect **=** *object***.TransitionEffect**

*object***.TransitionEffect =** *transitioneffect*

Legal values

The legal values vary by class.

Legal values for the Page class

Value	Description
$SSPageReplace	Instant replacement
$SSPageBottom	From bottom
$SSPageLeft	From left
$SSPageRight	From right
$SSPageBlinds	Blinds opening
$SSPageLouvers	Louvers opening
$SSPageBlocks	Block replacement
$SSPageCenter	From center out
$SSPageBoxIn	From outer box in
$SSPageZigZag	Zigzag replacement
$SSPageHorzln	Horizontal lines
$SSPageHVertln	Vertical lines
$SSPageTop	From top
$SSPageBoxOut	From inner box out
$SSPageHorizOut	Slide out horizontally
$SSPageVertOut	Slide out vertically
$SSPageFade	Fade to new
$SSPageDiagL	Diagonal left
$SSPageDiagR	Diagonal right
$SSPagePanL	Pan left
$SSPagePanR	Pan right
$SSPageScrollT	Scroll from the top
$SSPageScrollB	Scroll from the bottom
$SSPageDraw	Draw new page
$SSPageRain	Rain new page
$SSPagePBrush	Paintbrush new page
$SSPageShade	Shade new page
$SSPageCurtain	Open curtain
$SSPageBMPCol	Bitmap by colors

Legal values for the DrawObject class

Value	Description
$SSObjReplace	Instant replacement
$SSObjBottom	From bottom
$SSObjLeft	From left
$SSObjRight	From right
$SSObjBlinds	Blinds opening
$SSObjLouvers	Louvers opening
$SSObjBlocks	Block replacement
$SSObjCenter	From center out
$SSObjBoxIn	From outer box in
$SSObjZigZag	Zigzag replacement
$SSObjHorzln	Horizontal lines
$SSObjVertln	Vertical lines
$SSObjTop	From top
$SSObjBoxOut	From inner box out
$SSObjHorzOut	Slide out horizontally
$SSObjVertOut	Slide out vertically
$SSObjFade	Fade to new
$SSObjDiagL	Diagonal left
$SSObjDiagR	Diagonal right
$SSObjRain	Rain new page
$SSObjFlyL	Fly to the left
$SSObjFlyR	Fly to the right
$SSObjFlyU	Fly up
$SSObjFlyD	Fly down
$SSObjFlyLD	Fly to the left and down
$SSObjFlyRD	Fly to the right and down
$SSObjFlyLU	Fly to the left and up
$SSObjFlyRU	Fly to the right and up
$SSObjFlashS	Flash slow
$SSObjFlashM	Flash medium
$SSObjFlashF	Flash fast

```
' Example: TransitionEffect property
' Find out what the transition effect is for the current page, then
' change it to another transition effect.
Dim Effect As Variant
' Get the transition effect for the current page.
Effect = CurrentPage.TransitionEffect
' Set the new transition effect.
Effect = $SSPageBlinds
' Assign the new transition effect to the current page.
CurrentPage.TransitionEffect = Effect
```

Freelance Graphics: Type property

{button ,AL(`H_FLG_PLACEMENTBLOCK_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_TYPE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the "Click here..." block type.

Data type

Variant

Syntax

type **=** *placementblockobject***.Type**

*placementblockobject***.Type =** *type*

Legal values

Value	Description
pbTypeText	Click here to enter text
pbTypeSymbol	Click here to add a symbol
pbTypeChart	Click here to create a chart
pbTypeOrgChart	Click here to create an organization chart
pbTypeTable	Click here to create a table
pbTypeButton	Click here to add a button

```
' Example: Type property
' Find all of the placement blocks on a page, find out the type of
' placement block each is, then change all the placement blocks to
' symbol type placement blocks.
' For this example to work there must be at least one placement block
' on the page.
Dim pbType As Variant
Forall obj In CurrentPage.Objects
   ' Find all placement block objects.
   If (obj.IsPlacementBlock) Then
      ' Get the type of the placement block object.
      pbType = obj.Type
      ' Set the type of the placement block object to symbol.
      pbType = pbTypeSymbol
      ' Assign the new placement block type to the current
      ' placement block.
      obj.Type = pbType
   End If
End Forall
```

Freelance Graphics: Underline property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_UNDERLINE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the Underline attribute for the font. This underline style is a solid line under both words and spaces.

Data type

Integer (Boolean)

Syntax

value **=** *fontobject***.Underline**

*fontobject***.Underline =** *value*

Legal values

Value	Description
TRUE (non-0)	Underline on
FALSE (0)	Underline off

```
' Example: Underline property
' Give the specified text the underline attribute.
' For this example to work you must have a text object
' variable named MyText on the page.
MyText.TextBlock.Font.Underline = True
```

```
' Example: UndoEnabled property
' Enable undo functionality in Freelance Graphics.
CurrentApplication.Preferences.UndoEnabled = True
```

Freelance Graphics: UndoEnabled property
{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_UNDOENABLED_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the Undo enabled preference.

Data type
Integer (Boolean)

Syntax
flag **=** *preferencesobject***.UndoEnabled**

*preferencesobject***.UndoEnabled =** *flag*

Legal values

Value	Description
TRUE (non-0)	Undo command enabled
FALSE (0)	Undo command disabled

Freelance Graphics: UnitOfMeasure property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_UNITOFMEASURE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the unit of measure used in the application. In Freelance Graphics, this is always twips.

Data type

Variant

Syntax

unitofmeasure **=** *applicationobject***.UnitOfMeasure**

Legal values

The only legal value is $ltsScaleModeTwip.

```
' Example: UnitOfMeasure property
' Get the Freelance Graphics unit of measure.
Unit = CurrentApplication.UnitofMeasure
```

Freelance Graphics: UserClassNameApplication property

{button ,AL(`H_FLG_OLEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_USERCLASSNAMEAPPLICATION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) The Windows registry name of the application that corresponds to the OLE object.

Data type

String

Syntax

set *MyOLEName = OLEObject***.UserClassNameApplication**

Legal values

The string value of the following Windows registry key: \CLSID\<{class id}>\AuxuserType\3. To use WordPro as an example, the string value would be: "Lotus WordPro 97".

Usage

Use this property to find the name of the application corresponding to the OLE object that you have found.

```
' Example: UserClassNameApplication property
' This script searches through the current page for an OLE object,
' then determines if the object is a WordPro document.
' If it is, it puts up a message box.
ForAll Obj in CurrentPage.Objects
   If Obj.IsOLEObject then
      If Obj.UserClassNameApplication = "Lotus WordPro 97" then
         MessageBox "This is a WordPro document."
      End If
   End If
End ForAll
```

```
' Example: UserClassNameFull property
' This example searches through the current page for an OLE object,
' then puts the registry name up in a message box.
ForAll Obj in CurrentPage.Objects
   If Obj.IsOLEObject then
       Messagebox "This is the full registry name: " + Obj.UserClassNameFull
   End If
End ForAll
```

Freelance Graphics: UserClassNameFull property

{button ,AL(`;H_FLG_OLEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_USERCLASSNAMEFULL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) The full name of the OLE object as registered in the Windows registry.

Data type

String.

Syntax

set *MyOLEName = OLEObject***.UserClassNameFull**

Legal values

The string value of either of the following Windows registry keys: \CLSID\<{class id}>; or \<Prog Id>. To use WordPro as an example, the returned string would be: "WordPro.Document".

Usage

Use this property to find the full name of the OLE object as registered by the Windows registry.

Freelance Graphics: UserClassNameShort property

{button ,AL(`;H_FLG_OLEOBJECT_CLASS',0)} See list of classes

{button ,AL(`H_FLG_USERCLASSNAMESHORT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) The short name of the OLE object as registered in the Windows registry.

Data type

String.

Syntax

set *MyOLEName = OLEObject***.UserClassNameShort**

Legal Values

Returns the string that is in the Windows registry under the following key: \CLSID\<{class id}>\AuxUserType\2. To use WordPro as an example, the returned string would be "Document."

Usage

Use this property to discover or verify the type of embedded object you have found.

```
' Example: UserClassNameShort property
' This script searches through the current page for an OLE object,
' a WordPro document. If there is one, then it uses the native
' automation interface of the OLE object (WordPro, in this
' example) to make use of the script language of the OLE object.
ForAll Obj in CurrentPage.Objects
   If Obj.IsOLEObject then
      If Obj.UserClassNameShort = "Document" then
         Obj.Object.DocInfo
      End If
   End If
End ForAll
```

```
' Example: VersionID property
' Create a chart, find the version of the chart, then print that value
' in the output window
Dim value As Long
Dim cht As DrawObject
Set cht = CurrentPage.CreateChart
' Get the version of the chart
value = cht.VersionID
Print value
```

Freelance Graphics: VersionID property

{button ,AL(`H_FLG_BASEOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_VERSIONID_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determine the version of the code implementing an object. The version number changes for any release that is not 100% compatible with the previous release.

Data type

Long

Syntax

value **=** *object***.VersionID**

Legal values

Any numeric value.

```
' Example: VerticalAlignment property
' Align the specified text vertically.
' For this example to work you must have a text object
' variable named MyText on the page.
MyText.TextBlock.TextProperties.VerticalAlignment = $ltsAlignmentTop
```

Freelance Graphics: VerticalAlignment property

{button ,AL(`H_FLG_TEXTPROPERTIES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_VERTICALALIGNMENT_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the vertical alignment property for the text.

Data type

Variant (Enumerated)

Syntax

verticalalignment **=** *textobject***.VerticalAlignment**

*textobject***.VerticalAlignment =** *verticalalignment*

Legal values

Value	Description
$ltsAlignmentTop	Align at top
$ltsAlignmentVertCenter	Center
$ltsAlignmentBottom	Align at bottom

```
' Example: ViewMode property
' Find the current view mode, then change the view mode to Outliner.
Dim Mode As Variant
' Get the current view mode setting
Mode = CurrentDocument.ViewMode
' Change the view mode setting.
Mode = $ViewOutliner
' Assign the new view mode setting to the current document.
CurrentDocument.ViewMode = Mode
```

Freelance Graphics: ViewMode property

{button ,AL(`H_FLG_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_VIEWMODE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the view mode.

Data type

Variant (Enumerated)

Syntax

viewmode **=** *documentobject***.ViewMode**

*documentobject***.ViewMode =** *viewmode*

Legal values

Value	Description
$ViewDraw	Page view
$ViewOutliner	Outliner view
$ViewSorter	Page Sorter view
$ViewSlideShow	Screen Show view

Freelance Graphics: Visible property

{button ,AL(`H_FLG_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_VISIBLE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Get the visible attribute for the application.

Data type

Integer (Boolean)

Syntax

value = *applicationobject*.**Visible**

Legal values

Value	Description
TRUE (non-0)	Visible
FALSE (0)	Not visible

```
' Example: Visible property
' Find out if the current application is visible.
Dim value As Integer
value  = CurrentApplication.Visible
```

```
' Example: WaitForClick property
' Create a rectangle, get the WaitForClick value, change that value to one.
Dim value As Integer
Dim Obj As DrawObject
Set Obj = CurrentPage.CreateRect
' Get the current waitforclick value of the object.
value  = Obj.WaitForClick
' Change the waitforclick value.
value = 1
' Assign a new waitforclick value.
Obj.WaitForClick = value
```

Freelance Graphics: WaitForClick property

{button ,AL(`H_FLG_DRAWOBJECT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_WAITFORCLICK_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine if the object needs a mouse click in order to draw the object on the page during a screen show.

Data type

Integer (Boolean)

Syntax

value **=** *drawobject*.**WaitForClick**

drawobject.**WaitForClick =** *value*

Legal values

Value	Description
TRUE (non-0)	Wait for click to draw the object
FALSE (0)	Do not wait for click to draw the object

```
' Example: Width property
' Make the border width of the specified rectangle thin.
' You must have a rectangle object variable named MyRect on the page.
MyRect.Border.Width = $ltsBorderWidthThin
```

Freelance Graphics: Width property

{button ,AL(`H_FLG_APPLICATIONWINDOW_CLASS;H_FLG_DOCWINDOW_CLASS;H_FLG_DRAWOBJECT_CLASS;H_FLG_LINESTYLE_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_WIDTH_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the width of a border or line.

(Read-only) Get the width of the application window, document window, or drawn object.

Data type

Integer (ApplicationWindow and DocWindow classes)

Variant (Border and LineStyle classes)

Long (DrawObject class)

Syntax

width **=** *object***.Width**

*object***.Width =** *width*

Legal values

For ApplicationWindow, DocWindow, and DrawObject classes: any positive value representing the width in twips.

For Border and LineStyle classes:

Value	Description
$ltsBorderWidthNone	None
$ltsBorderWidthVeryThin	Thinnest
$ltsBorderWidthThin	
$ltsBorderWidthModeratelyThin	
$ltsBorderWidthMedium	
$ltsBorderWidthModeratelyThick	
$ltsBorderWidthThick	
$ltsBorderWidthVeryThick	
$ltsBorderWidthExtremelyThick	Thickest

Freelance Graphics: WordDoubleUnderline property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_WORDDOUBLEUNDERLINE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the Word double underline attribute for the font. This is a double line under words but not spaces.

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer (Boolean)

Syntax

flag **=** *fontobject*.**WordDoubleUnderline**

fontobject.**WordDoubleUnderline =** *flag*

Legal values

Value	**Description**
TRUE (non-0)	WordDoubleUnderline on
FALSE (0)	WordDoubleUnderline off

```
' Example: WordDoubleUnderline property
' Ignored by Freelance Graphics; it is here for compatibility reasons only.
```

Freelance Graphics: WordUnderline property

{button ,AL(`H_FLG_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_WORDUNDERLINE_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Determine the Word underline attribute for the font. This is a single line under words but not spaces.

Note This property is ignored by Freelance Graphics, and is provided for compatibility with other Lotus products.

Data type

Integer (Boolean)

Syntax

flag **=** *fontobject***.WordUnderline**

*fontobject***.WordUnderline =** *flag*

Legal values

Value	Description
TRUE (non-0)	WordUnderline on
FALSE (0)	WordUnderline off

```
' Example: WordUnderline property
' Ignored by Freelance Graphics; it is here for compatibility reasons only.
```

Freelance Graphics: WorkDir property

{button ,AL(`H_FLG_PREFERENCES_CLASS;',0)} See list of classes

{button ,AL(`H_FLG_WORKDIR_PROPERTY_EXSCRIPT',1)} See example

(Read-write) Get or set the work directory preference.

Data type

String

Syntax

workdirectory **=** *preferencesobject***.WorkDir**

*preferencesobject***.WorkDir =** *workdirectory*

Legal values

Any directory.

```
' Example: WorkDir property
' Specify the work directory--in this example the directory
' is set to c:\lotus\work\flg.
CurrentApplication.Preferences.WorkDir = "c:\lotus\work\flg"
```

Help is not available

The topic you selected is in a Help file that is not available.

To install additional Help files, run the Install program, select the Customize features - Manual install option, and specify the Help file(s) you want to install.

Error handling in Freelance Graphics

Freelance Graphics allows you to use the standard LotusScript "On Error" mechanism to trap and handle Freelance Graphics errors. When a script traps Freelance Graphics errors, the errors are not displayed by Freelance Graphics. You can either use a script to display a message or you can completely ignore the error once you trap it.

Error Numbers

Freelance Graphics does not provide a separate number for every possible error condition, instead it reports a few "categories" of errors to LotusScript. They are as follows:

FLGErrCategoryError	= 9001
FLGErrCategoryWarning	= 9002
FLGErrCategoryInformation	= 9003
FLGErrCategoryTip	= 9004
FLGErrNotSupported	= 9000

These ID's are defined as constants in the file "FLGERR.LSS", which is installed in the \LOTUS\FLG directory. You can %INCLUDE this file in your scripts and use the constant names to identify error categories for better readability.

Most Freelance Graphics messages fall into the "Error" category. These will appear with the error number 9001. The other categories are provided in case you prefer to handle some errors but not to handle other messages such as Tips. You can either choose to trap only what you are most concerned with, or, you can trap all errors and handle each of them differently.

For example, if you want to trap only one type of error, use this script as model:

```
On Error 9001 Goto FreelanceError
```

If you want to trap other messages in addition to errors, use this script as a model:

```
If Err = FLGErrCategoryError
MsgBox "Error"    'tell the user
Resume Next
Else If Err = FLGErrCategoryTip
Resume Next     'just ignore it
End If
```

Note The range of numbers Freelance Graphic uses (9000 - 9999) is reserved for it alone to avoid SmartSuite error number conflicts with other Lotus applications or with LotusScript itself.

Error strings and messages

Although Freelance Graphics has a small number of error numbers (categories), the error string that is passed to a script matches exactly what Freelance Graphics would have displayed to the user. In the following example, when "MySub" runs, it shows error 9001, as would many Freelance Graphics errors, but the actual error text contains a very detailed message specific to the error condition.

```
Sub MySub
On Error Goto ProcessError
MessageBox "Start of Script"
CurrentPage.CreateSymbol "doesnotexist", 2
Goto Done

ProcessError:
MessageBox "Found Error "& Str(Err) & ": " & Error$,MB_OK, "My Error Dialog"
Resume Next

Done:
End Sub
```

Note The routine MySub sets trapping on for all errors and messages. In this subrouteen, the script purposely generates a "file not found" error (category 9001) in Freelance Graphics, which is then passed to the subrouteen (and not displayed by Freelance Graphics). The message that this specific error generates is displayed using the standard

"Err" and "Error$" keywords (see LotusScript Language Help for more about "Err" and "Error$"), then the subrouteen resumes at the next line.

Limitations

If a message or error normally presents the user with a decision such as Yes/No or Ok/Cancel, that error will be displayed to the user for their input, it will not be trapped by the script. Also, some errors displayed during creation or editing of charts may not be trapped.

Overview: Scripting in Freelance Graphics

The Application Programming Interface (API) for Freelance Graphics functionality is an extension of LotusScript--a cross-product Basic scripting language

You can use LotusScript to automate tasks the user does on a regular basis in Freelance Graphics. For example, you can use LotusScript and the Freelance API to create applications that make it easy to edit a frequently used presentation (by attaching scripts to a SmartMaster). You can also write scripts that:

- Attach to a page, a "Click here" block, or SmartIcons
- Operate on documents, pages, and objects
- Make presentations using information gathered from various Lotus products, such as 1-2-3.

With the Freelance Graphics API, you can create "scripts" that execute when you click a button, an icon, or a "Click here" block. You can also execute a script when you choose Edit - Script - Run.

You use the Script Editor and Debugger to create, edit, and debug scripts.

For more information, scroll down or click one of the following:

Sources of information

There is information in Help about the Freelance Graphics API (look under Help - Help Topics - LotusScript). There is also context-sensitive Help for the Script Editor and Debugger and for the overall LotusScript language.

To get printed documentation, see Ordering the LotusScript Documentation Set.

The structure of LotusScript

The API and LotusScript use the principles of object-oriented programming. For information about object-oriented programming and LotusScript in Lotus applications, see the *LotusScript Programmer's Guide*.

Classes

The Freelance Graphics API consists of approximately 27 classes (custom LotusScript data types). Each class definition includes a set of properties and methods. A small number of classes also have events, that is actions that run scripts.

When you use the Freelance Graphics LotusScript API, a document, for example, is an instance of the Document class. Individual elements on a page, such as a rectangle or a text block, are instances of the DrawObject class. For a list of classes, as well as reference information, see Freelance Graphics LotusScript Classes A-Z.

Most Freelance Graphics classes are derived from the BaseObject class. Since the BaseObject class is an abstract class, you never create instances, or objects, of that class.

Some classes are derrived from the DrawObject class and inherit properties and methods from it, such as the Selection class and the PlacementBlock class.

Collection classes

The collection classes in Freelance Graphics (Documents, Pages, Objects, and Colors) inherit from the BaseObject class. They are collections of objects.

The following table shows what indexes the collection classes use.

Class	Indexable by
Documents	Number

Pages	Number and name
Objects	Number
Colors	Name

You can get the index of any object by using the GetIndex property. The index number is by priority. Note that the current document is always one. For information about collection classes, see the *LotusScript Programmer's Guide*.

Note In Freelance Graphics script syntax, the index of the first item in a collection or table is one. In other Lotus applications (WordPro and Approach, for example), the index of the first item is zero. In the future, script indexing may be standardized across all Lotus applications. If zero becomes the standard for the first collection or table item, you will need to adjust all existing Freelance Graphics script statements containing collection or table references. You can use OPTION BASE to change the indexing base. See the *LotusScript Language Reference* for more information or online the LotusScript Reference.

Class containment heirarchy

At the top of the heirarchy is the Freelance Graphics application. It contains the Documents class, a collection class representing all of the documents currently open in the application, and a Document class, representing individual presentations (.PRZ). Each Document class contains a Pages class, a collection class representing all of the pages in a document, and a Page class, representing an individual page. The Page class contains the DrawObject class, representing an element on the page, such as a "Click here" block, a chart, or an OLE object.

Properties

Properties define the appearance and behavior of objects. Many object classes have properties defining visual attributes, such as background color, size, and location. Some properties apply to only one object class. For example, the Title property is unique to the Page class. On the other hand, the Width property is a property of the AppWindow, Border, DocWindow, and DrawObject classes.

Some classes may also act as a property for another class. For example, an instance of the Color class can be a property for an instance of the Font class. When a class is a property, the property description tells you.

For a list of properties and reference information about them, see Freelance Graphics LotusScript Properties A-Z.

Methods

Methods are subprograms you use to manipulate objects. For example, you can use the Move method to move an instance of the DrawObject class (a rectangle, for example), or you can use the Copy method to copy an instance of the DrawObject class.

For a list of methods and reference information about them, see Freelance Graphics LotusScript Methods A-Z.

Events

Events are scriptable actions that are associated with certain classes. For example, you can use the PlacementBlock class Click event to run a script when a placement block is clicked.

For a list of events and reference information about them, see Freelance Graphics LotusScript Events A-Z.

Freelance Graphics API rules

Using names

Names offer a convenient way of manipulating objects and elements in the Freelance Graphics API. In Freelance Graphics every object of the Application class, Document class, Page class, and DrawObject class has a default name (however, objects of the Font, Background, Border, and LineStyle classes, and objects of the chart classes do not have default names).

A Document object's name is the file name and is read-only. A DrawObject object has a default descriptive name (such as, PlacementBlock1 or Rectangle1) which can be changed through a script. The name of the page is shown in the Infobox or the IDE; it can be changed by editing the name in the InfoBox or by writing a script. Page 1 is the default name of the first page created in a presentation, Page 2 is the name of the next page, and so on.

The names of all objects with events are listed in the IDE. However, the actual names of the elements (default or otherwise) are not listed in the IDE. To find out the names of all the elements on a page that you can manipulate with a script, run the following script:

```
'Find the names of all objects on this page.
ForAll Objs in CurrentPage.Objects
   Print Obj.Name
End ForAll
```

Note Print output appears in the Output window of the IDE.

So, for example, if you have several elements on a page (a text block, a rectangle, and an elipse), and you want to manipulate the rectangle, you would use the script described above and learn that the rectangle's name is Rectange1. Once you know the name of an element, you can manipulate it. If you want to move the rectangle, write a script such as the following:

```
Set Rect = CurrentPage.FindObject("Rectangle1")
Rect.move 1000, 1000
```

You can even change the element's name. For example:

```
Rect.name = "Euclid"
```

Note Once you change the name of an element, you must refer to it by the new name. In the above example, once you rename "Rectangle1" to "Euclid," you have to refer to the rectangle as "Euclid" the next time you used the FindObject method, unless, of course, you change the name again.

Freelance Graphics stores the names given to pages and elements in the presentation so that they are available in future script sessions. You can find a drawn object if you know its name by using the FindObject method, as demonstrated above. For more information, see FindObject method.

You can use Bind to bind to an instance of the Document, Page, and DrawObject classes. For example:

```
Dim p As Page
Set p = Bind("Page 1")
Print p.number
```

Predefined global variables
Predefined variables let you operate on Freelance Graphics objects:

Predefined Global Variable	Description
CurrentApplication	The current session of Freelance Graphics. Uses the properties and methods of the Application class
CurrentApplicationWindow	The application window of the current session of Freelance Graphics. Uses the properties and methods of the ApplicationWindow class
CurrentDocument	The current Freelance Graphics document. Uses the properties and methods of the Document class.
CurrentDocWindow	The current Freelance Graphics document window. Uses the properties and methods of the DocWindow class
CurrentPage	The current page. Uses the properties and methods of the Page class
Selection	The currently selected object(s). Uses the properties and methods of the DrawObject class

For examples of how to use Global variables, see Using LotusScript predefined variables in Overview:Creating, editing, and debugging a script.

Running a script

You can attach scripts to a document, a page, a "Click here" block, or SmartIcons, and that run when the user performs some action. Or, you can write scripts that run when you choose Edit - Script - Run, or are attached to an icon. These are different processes.

To attach (or assign) a script to a page or a "Click here" block, you must be in a content topic (see To open a content topic). If the script is attached to a "Click here" block or a page, it is saved in the content topic itself. The script runs automatically when the user performs some action, such as opening the page or clicking the "Click here" block. The action depends on which event you choose to use to trigger running the script.

See
Attaching a script to a "Click here" block,
Attaching a script to a content page,
Creating a script button.

When you want to run a script by choosing Edit - Script - Run, you you first must have saved the script as an .LSS or. LSO file (a compiled LSS file). You run the script by choosing Edit - Script - Run, and typing the .LSS (or .LSO) file name.

For information about attaching scripts to SmartIcons, see Attaching a script to an icon.

For information about creating, editing, and debugging a script, see Overview: Creating, editing, and debugging a script.

{button ,AL(`H_FLW_SCRIPT_RUN_STEPS;H_FLW_SCRIPT_CR_ED_DEB_OVER;H_FLG_SCRIPT_ERROR_HANDLING_OVER;H_SCRIPTING_TIPS_FOR_FREELANCE_GRAPHICS_OVER;H_FLW_SCRIPT_UPGRD_OVER;',0)} See related topics

Overview: Creating, editing, and debugging a script

The Script Editor and Debugger in which you write, run, and debug a script has Help. Press F1 when you want information. To launch the Integrated Development Environment (IDE) in Freelance Graphics, choose Edit - Script - Show Script Editor. For more information about working in the Script Editor and Debugger, look under Script Editor or Script Debugger in the Freelance Graphics Help index.

Tip You can also review the list of available Freelance Graphics classes, properties, methods, and events by using the browser in the IDE. You can get help on a highlighted item by pressing F1.

For more information, scroll down or click one of the following:

Creating objects and assigning object references

You can access Freelance Graphics objects, including drawn objects, pages, and whole presentations, by assigning a reference to that object. References can be assigned to existing objects or to objects that are created within a script.

To assign an existing Freelance Graphics object a reference variable, use the Set statement. Set must be used any time a reference variable is assigned to an instance of a class. For example:

```
Dim MyPage As Page
Dim MyRect As DrawObject

' Set MyPage equal to the third Page in the presentation and MyRect
' equal to the second item on MyPage.
Set MyPage = CurrentDocument.Pages.Item(3)
Set MyRect = MyPage.Objects.Item(2)

' Set MyRect equal to the object named My Rectangle.
Set MyRect = CurrentPage.FindObject("My Rectangle")
```

To create a Freelance Graphics object, you must use the appropriate method. In general, to create an object follow the method available to the class. For example, the methods for creating drawn objects belong to the Page class. To create a rectangle:

```
Dim MyRect As DrawObject
Set MyRect = CurrentPage.CreateRect (2000, 3000, 3000, 4000)
```

The method for creating a page belongs to the Document class:

```
Dim MyPage As Page
Set MyPage = CurrentDocument.CreatePage("Title of Page", 2)
```

The method for creating a document, NewDocument, belongs to the Application class. For example:

```
Set MyDoc = CurrentApplication.NewDocument("test.prz")
```

Using LotusScript predefined product variables

You can use the predefined global variables to write scripts that operate on the currently selected element, page, document, document window, application window, or application.

Selection is a global variable that represents the currently selected element or elements on a page. To change the pattern of the currently selected element or elements on the page:

```
Sub Main
   Selection.Background.Pattern=$LtsFillGray2
End Sub
```

CurrentPage represents the current page and uses the properties and methods of the Page class. To delay a page transition on the current page by ten seconds:

```
Sub Main
   CurrentPage.PageTransitionDelay=10
End Sub
```

Using global product variables to assign object variables
Use global product variables as a convenient way to access all other objects. Freelance Graphics always maintains valid values for you.

The Set statement must be used any time an object variable is assigned to an instance of a class. Also, the Pages and Objects classes are collections that can be indexed (as in an array). For example:

```
Dim MyRect As DrawObject
Dim MyPage As Page
' Set MyPage equal to the third page in the presentation,
' and MyRect to the second item on MyPage.
Set MyPage = CurrentDocument.Pages(3)
Set MyRect = MyPage.Objects(2)
```

Notice that in the example the predefined global variable, CurrentDocument, is used to refer to the current document and that the collection property, Pages, is used to refer to the third page of the document. Once you use the global variable, CurrentDocument, to assign the object variable, MyPage, the example shows how to use MyPage to refer to an explict element on a page.

You can access Freelance Graphics elements, including   pages and whole presentations, by assigning a reference to that element, page, or presentation. For example, you can assign references to elements that are created within a script or to already existing elements. In the example that follows, note the use of the global variable CurrentPage in combination with the CreateRect method to assign the object variable, Rect1:

```
Dim Rect1 As DrawObject
' Create a rectangle of default size, then name it MyRect1.
Set Rect1 = CurrentPage.CreateRect
Rect1.Name = "MyRect1"
```

Later in a script you could use the name, MyRect1, that you gave to the rectangle in the above code example, to find the rectangle so that you can manipulate it in some way. You can find an existing named element (for example, MyRect1) on the current page by using the global variable CurrentPage and the FindObject method (a Page Class method that CurrentPage can use). For example:

```
Dim Obj1 As DrawObject
' Find a rectangle named MyRect1.
Set Obj1 = CurrentPage.FindObject("MyRect1")
```

To continue with this example, to move the rectangle once you find it, use the following line of code. The code makes use of the object variable Obj1 that was set above and the Move method (a method of the DrawObject class). Obj1 is an instance of DrawObject.

```
Obj1.Move 1000,1000
```

Creating objects
To create a Freelance Graphics element, you must use the appropriate method. In general, you create an element by using a method of the appropriate class. You have already seen some examples of creating in the previous section, "Using global product variables to assign object variables." In this section, "Creating objects," you will find more examples.

The methods for creating elements on a page belong to the Page class. For example, to create a rectangle:

```
Dim MyRect As DrawObject
Set MyRect = CurrentPage.CreateRect (2000, 3000, 3000, 4000)
```

In this example, you create a rectangle, MyRect, by using the global variable CurrentPage (it uses the methods and properties of the Page class) and the CreateRect method (a Page class method) to create the rectangle. The numbers in parenthesis give the location, size, and width of the rectangle on the page. See Freelance Graphics Reference Help for more information about how to use the CreateRect method.

Because the method for creating a page belongs to the Document class, you can use it with the global variable CurrentDocument. In the following example the method, CreatePage, takes two parameters: the title of the page and the SmartLook (or template) that the page will be based on.

```
Dim MyPage As Page
Set MyPage = CurrentDocument.CreatePage("Title of Page", 2)
```

As part of the creation process, elements and pages are given names by default. These names can be changed by scripts. Freelance Graphics stores the names given to pages and elements in the presentation, so that they are available in future script sessions. You can assign a variable that references an existing page in the following way:

```
Dim Pg As Page
Set Pg = CurrentDocument.Pages("Page 1")
```

Note "Page 1" is the default name of the first page created in the presentation, "Page 2" is the name of the next page, and so on.

To change the name of a page using a script, do the following (continuing with the above example):

```
Pg.Name = "Agenda"
```

The NewDocument method for creating a document belongs to the Application class. The following script creates a new document.

```
CurrentApplication.NewDocument
```

Note You must save a document to name it.

Also, the OpenDocument method opens an existing document (in that sense it "creates" a Document object). For example, to open an exisitng presentation (PROPOSAL.PRZ) do the following:

```
Set MyDoc = CurrentApplication.OpenDocument("proposal.prz")
```

You use these methods to open or create presentations.

Running a script from the command line

You can run a script by typing

```
C:\Freelancepath\F32main /r lsscript.lss filename.prz
```

from the command line.

(*lsscript.lss* is the name of the script you want to run, and *filename* is the name of the presentation you want to run the script in.)

- In Windows 95, click Start in the taskbar, choose Run, and type the command.
- In Windows NT, Press CTRL+ESC to display the task list, type the command in the New Task edit box, then click the Run button.

Attaching scripts in SmartMaster content files

The advantage of attaching scripts in an .SMC file is that .SMC files are the "templates" for presentation (.PRZ) files. Scripts attached to .SMC files can be used each time you create a presentation using the .SMC file. However, scripts attached to a .PRZ file can only be used in that .PRZ file. Generally speaking, you attach scripts to events. For more information on content topics, see Overview: What is a content topic and Overview: Ways to create your own content topics.

Using Events

An event is an action associated with a given class, such as the Click event for the PlacementBlock class. You attach a script to an event. When the event occurs, the script runs. There are events associated with the Document class, the Page class, and the Placement block class. For example, the Page class has two events: Activated and Created.

The PlacementBlock class event is Click. Placement blocks, that is, "Click here" blocks, can, therefore, run a script when the user clicks the placement block.

Placement blocks or "Click here" blocks

A "Click here" block, also known in scripting as a placement block, can be either a TextPlacementBlock, a Button, a SymbolPlacementBlock, a ChartPlacementBlock, an OrgChartPlacementBlock, a TablePlacementBlock, or a DiagramPlacementBlock. Once created, all of these placement blocks can have scripts attached to them. You can create placement blocks only while:

- Editing SmartMaster content files (.SMC files)
- Editing SmartMaster look files (.MAS files)
- Editing a page layout or backdrop in a .PRZ file

For information about creating and attaching a script to a "Click here" block, see Attaching a script to a "Click here" block.

Attaching scripts to icons

You can attach scripts to icons. For more information on icons, see Attaching a script to an icon.

Trapping errors

For information about how to handle errors in Freelance Graphics scripts, see Error handling in Freelance Graphics

Sample scripts

For examples of working code and of the object-oriented syntax used in the Freelance Graphics API, review the scripts that are used by SmartMaster content (.SMC) files in Freelance Graphics. Scripts in these files refer to the source code contained in the file GTSCRPT.LSS, located in the \LOTUS\SMASTERS\FLG directory.

Caution Modifying script code in this file may cause problems with Freelance Graphics content topics. Make a copy of the file to experiment with.

{button ,AL(`H_AV_ATTACHING_A_SCRIPT_TO_AN_ICON_STEPS;H_FLW_SCRIPT_IDE_STEPS;H_FLW_SCRIPT_OVER;H_FLW_SCRIPT_RUN_STEPS;H_SMDESIGN_SCRIPT_BUTTON_STEPS;H_FLG_SCRIPT_ERROR_HANDLING_OVER;H_SCRIPTING_TIPS_FOR_FREELANCE_GRAPHICS_OVER',0)} See related topics

Opening the Script Dialog Editor

To open the Script Dialog Editor, choose Edit - Script - Show Dialog Editor.

Opening the Script Editor and Debugger
To open the Script Editor and Debugger, choose Edit - Script - Show Script Editor.

{button ,AL(`H_FLW_SCRIPT_OVER;H_FLW_SCRIPT_RUN_STEPS',0)} See related topics

Running a script

Follow these steps to run a script that is not attached to a page or a "Click here" block.

1. Choose Edit - Script - Run.
2. Type the script file name in the File name box.
3. Click Open.

Note To see error messages and output, open the Output window; choose Edit - Script - Show Output Window.

{button ,AL(`H_FLW_SCRIPT_IDE_STEPS;H_FLW_SCRIPT_OVER;H_FLG_SCRIPT_ERROR_HANDLING_OVER', 0)} See related topics

Script information for upgraders

This topic contains information about new commands and capabilities in the Freelance Graphic script API.

What's new

Freelance Graphics now makes use of error handling capabilities.   See Error handling in Freelance Graphics for complete information.

There is a new method for the Document class, see Export method for more information.

FAQ (Frequently Asked Questions)

Here are some tips and scripts for Freelance Graphics covering common tasks , general issues , and troubleshooting.

For more information, scroll down or click one of the following:

Common Tasks

How do I create a placement block (that is, a button or a "Click here" block)?
You must be editing a SmartMaster with content file (that is, an .SMC file), a SmartMaster look file (an .MAS file), or a page layout or backdrop in a .PRZ file. To create a button or a "Click here" block, choose Create - "Click here" Block. You can then attach scripts to the "Click here" block by making use of the placement block Click event.

Note There are several instances of the PlacementBlock class in Freelance Graphics: TextPlacementBlock, SymbolPlacementBlock, ChartPlacementBlock, OrgChartPlacementBlock, TablePlacementBlock, DiagramPlacementBlock, and Button. You can specify the type when you create the placement block; TextPlacementBlock is the default. Do not confuse the TextPlacementBlock with the TextBlock--they are different. A TextBlock does not have an event associated with it, and a script cannot be attached to it.

How do I attach a script to a placement block (that is, a button or a "Click here" block)?
You must be editing a SmartMaster with content (that is, an .SMC file) or a presentation file (.PRZ) that already has "Click here" blocks or buttons. Open the IDE by choosing Edit - Script - Open Editor. If you are editing an .SMC file, you can open the InfoBox for the button or "Click here" block, click the Basics tab and open the IDE from there using the Edit/Create button. Then you use the placement block Click event to write a script that runs when the user clicks the "Click here" block.

In short, you can attach a script to any element that has an event associated with it: a placement block, a page, or a document.

How do I create a new presentation?

You can use the NewDocument method to create a new Freelance Graphics presentation. The following example creates a new presentation in a separate window, leaving the current presentation still open in its own window.

```
CurrentApplication.NewDocument "MyPresentation", , "buttons.mas"
```

How do I copy and paste an object?
You can use the Replicate method or the Copy and Paste method to create a duplicate of an element:

```
Sub MakeReplicate
   Dim MyRectangle As DrawObject
   Dim NewCopy As DrawObject
   ' Assign object variables, create a rectangle and make a replicate of it.
   MyRectangle = CurrentPage.CreateRect
   Set NewCopy = MyRectangle.Replicate
End Sub
```

You can use the Copy and Paste methods with any object or group of objects on the page. For example:

```
Sub ClipboardDemo
   Dim MyRectangle As DrawObject
   ' Make a new rectangle
   MyRectangle = CurrentPage.CreateRect
   ' Copy the rectangle to the clipboard
   MyRectangle.Copy
   ' Pastes clipboard contents onto the page
   Set NewObj = CurrentPage.Paste
   End Sub
```

How do I change the view?
You can change between Current Page view, Outline view, or the Sorter view:

```
CurrentDocument.ViewMode = $ViewOutliner
CurrentDocument.ViewMode = $ViewSorter
CurrentDocument.ViewMode = $ViewDraw
```

Most product commands are available from Current Page view, and Current Page view should be used for most scripts.

Note Many script commands do not alter the appearance of the screen. So, while a script runs, you may not see any action taking place.

How do I add pages?
You can use the CreatePage or CreatePageFromTemplate (when there is an active Content SmartMaster) commands to add new pages. Either of the following examples will work.

Example 1:

```
   ' Add a title page
   CurrentDocument.CreatePage "My title page", 1


   ' Or, you can add a title page from the current Content SmartMaster.
   CurrentDocument.CreatePageFromTemplate "A Content SmartMaster " +_
      "title page", 1
```

Example 2:

In this example, the object variable, MyPage, is identified with the page that is created.

```
   ' Add a title page and assign an object variable to the new page.
   Set MyPage = CurrentDocument.CreatePage ("My title page", 1)


   ' Or, you can add a title page from the current content SmartMaster
   Set MyPage = CurrentDocument.CreatePageFromTemplate _
      ("A Content SmartMaster page", 1)
```

How do I change the look of a presentation?
You can change the look of the current presentation by setting the SmartLook property:

```
' Change the look to the Buttons SmartMaster set.
CurrentDocument.SmartLook = "buttons.mas"
```

How do I add a block of text to a page?
As in the user interface, there are two ways to add text to a presentation page with LotusScript.

- Create a new text object on the page
- Fill in an existing text block.

To create a new text object on the current page, use the CreateText command as follows:

```
Set MyTextBlock = CurrentPage.CreateText
MyTextBlock.Text = "Here's a line of text"
```

There are three ways to access an element iin a presentation. The following examples use each of these three ways to access an existing text block:

1. You can assign a DrawObject variable to the text block:

   ```
   Set MyClickHere = Bind DrawObject("Text2")
   ' Use the variable and the Text method to put text into the object.
   MyClickHere.Text = "Here is a line of text"
   ```

2. If the desired text block is selected, you can access it through the current selection:

   ```
   Selection.Textblock.Text = "Here is a line of text."
   ```

3. You can cycle through the objects on the current page and find a "Click here" block (that is, a placement block) and then put text in it. Note, when you put text in a placement block, it becomes a text block and it is no longer recognized as a placement block.

   ```
   ForAll obj in CurrentPage.Objects
      If obj.IsPlacementBlock Then
         If obj.Type = $pbTypeText then
            obj.Text = "Here's a line of text."
         End If
      End If
   End ForAll
   ```

How do I add clip art?
You can use the CreateSymbol command to add existing clip art or diagrams to the current page.

```
' Add the 3rd animal clip art drawing
CreateSymbol "animals", 3
```

or

```
Set MySymbol = CreateSymbol ("animals", 3)
```

To place a symbol in an existing placement block, you can cycle through all of the elements on the current page until you find a placement block object.

```
ForAll obj in CurrentPage.Objects
   If obj.IsPlacementBlock Then
      If obj.PromptText = "Click here to add clip art" Then
         obj.insert CreateSymbol ("animals", 3)
      End If
   End If
End ForAll
```

How do I rearrange the pages of a presentation?
You can move pages by setting their page number.

```
ThisPage = CurrentPage.Number
CurrentPage.Number = ThisPage + 2
```

How do I print a presentation?
The Print method prints the active presentation to the current printer.

```
CurrentDocument.Print
```

How do I run a presentation as a screen show?
Under LotusScript, a screen show is another view mode. You can start a presentation as a screen show from script by changing the ViewMode property.

```
CurrentDocument.ViewMode = $ViewSlideShow
```

General issues

Is there a list of all classes, properties, methods, and events I can access in a presentation?
For a complete list of classes, properties, methods, and events, with links to reference information, see Freelance Graphics LotusScript A-Z. You can also see a list of properties, methods, and events for each class in the IDE.

How do I access text in a TextBlock?
Here are two examples of how to access a text block.:

1. You can assign a DrawObject variable to the text block and then alter the text.

```
Set MyClickHere = Bind DrawObject("Subtitle")
MyClickHere.TextBlock.Text = "Here's a line of text"
```

Note You can use Bind to bind to instances of the Document, Page, and DrawObject classes.

2. If the desired text block is selected, you can access it by using the current selection:

```
Set MyClickHere = Selection.Textblock
```

How do I add multiple lines of text to a text block?
To convert lines of text from another source to a text block, you first convert the text to Freelance Graphics markup format, which uses the characters "<=" in place of carriage return and line feed characters.

The following function converts a Notes-formatted string to a markup-formatted string:

```
Function ConvStrForFLG (Str1 As String) As String
   ' Converts a Notes string to a string that FLG can use --
   ' i.e. converts the carriage return / line feed
   ' (chr(13) / chr(10)) to equivalent markup characters ("<=").
   '   Inputs:  Str1 - string to convert
   '   Outputs: none
   '   Returns: converted string in markup form

   Dim pStr1, pStr2 As Integer

   ConvStrForFLG$ = ""
   pStr1 = 1
   pStr2 = Instr (Str1, Chr$(13))

   ' Convert embedded CR/LF
   Do While pStr2 <> 0
      ConvStrForFLG$ =  ConvStrForFLG$ + _
         Mid$ (Str1, pStr1, pStr2 - pStr1) + "<="
      pStr1 = pStr2 + 2
      pStr2 = Instr(pStr1, str1, Chr$(13))
   Loop

   ' Handle last part of string
   If Len (Mid$(Str1, pStr1)) <> 0 Then
      ConvStrForFLG$ = ConvStrForFLG$ + Mid$(Str1, pStr1)
   End If
End Function
```

How do I search and replace text?
LotusScript in Freelance Graphics provides no direct methods for searching and replacing text. However, search and replace functions are available to the user under the Freelance Graphics Edit menu.

You can write a script that searches the document for text blocks, then extracts text strings, and then compares each string member with the text you want to replace or find.

Error handling
To find out complete details on how to trap errors and handle messages, see Error handling in Freelance Graphics

Troubleshooting

Why doesn't my attached script run when I click the button?
There are two possibilities:

1. If a placement block has been filled in with text, a chart, and so on, it is no longer identified as a placement block.

Any script that operates on that placement block will no longer recognize it as a placement block and the script will fail to execute. For example, if you are searching for a placement block, but all there is on the page is a "Click here" block that is now filled in with text, then using IsPlacementBlock will not recognize any placement block on the page. On the other hand, IsTextBlock will recognize the block.

Note If you delete the text that was entered into a "Click here" block, it is recognized as a placement block again.

2. If a placement block is renamed in the course of a script's execution or by user intervention, the changed name will cause a script that looks for that placement block using the old name to fail. For example, if "TextPlacementblock1" gets renamed "My block," the script that works on that placement block would have to take the name change into account, or the script will fail.

Can I create multiple buttons with attached scripts on a page?
Yes, multiple buttons are accessible from a SmartMaster content (.SMC) file and from a SmartMaster looks (.MAS) file.

Ordering the LotusScript Documentation Set

These books are available in the CD-ROM version of your SmartSuite package as Online Books, available for viewing on Lotus' World Wide Web site (http://www.lotus.com/home.nsf/welcome/suitedev), and available in printed form in the SmartSuite Application Developer's Documentation Set. These books are available only in English.

- *LotusScript Language Reference* provides a comprehensive summary of conventions and basic commands for the LotusScript language. *LotusScript Language Reference* provides the foundation for programming any product that supports the LotusScript programming language.
- *The LotusScript Programmer's Guide* describes the basic building blocks for LotusScript applications and provides many working examples.

You can receive one *complimentary* copy of the SmartSuite LotusScript Documentation Set. To take advantage of this offer, fill out the form below and mail it to the appropriate office, postmarked no later than December 31, 1998. The only additional charge you are responsible for is the shipping and handling fee. US customers can pay by check (payable to Lotus Development Corporation and drawn on a US bank) or credit card; others must pay by credit card. Please allow 4 - 6 weeks for delivery.

For details on mailing addresses and shipping and handling charges, see the online Fulfillment Information.

Name ____

Company ____

Address 1 (no PO boxes please) ____

Address 2 ____

City ________________________ State/Province _____________________________

Zip/Postal Code ___________ Phone (____) _____ - _________ ext. ______

Shipping and handling charge: ___________

Payment Method: Visa / Mastercard / Amex / Check (circle one)

Card Number ________________________ Exp Date ________ / _____

Signature ____

Overview: Fulfillment Information

Refer to the following table when ordering the Application Developer's Documentation Set for SmartSuite.

- Identify the fulfillment center/centre for your country
- Copy the mailing address to the front of an envelope coupon
- Insert payment for shipping and handling charges and the order form into the envelope

Use the appropriate shipping and handling charge for your country

The following charges include VAT (Value Added Tax) where applicable.

Country	Currency	Charge	Fulfillment Center Mailing address
Australia	Australian Dollar	A$35	Lotus Development Pty Ltd Customer Service Department Level 12, 321 Kent Street Sydney NSW 2000 Australia
Austria	Austrian Schilling	300 OS	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Belgium	Belgian Franc	900BF	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Canada	Canadian Dollar	C$15	Lotus Development Corporation SmartSuite Documentation P.O. Box 670 Scarborough, Ontario M1K 5C5
Denmark	Danish Krone	Dkr 175	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Finland	Markka	130 mk	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
France	French Franc	150F	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Germany	Deutchmark	45 DM	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland

Ireland	Punt	IR£15	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Italy	Lira	L. 4500	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Netherlands	Guilder	F 50	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
New Zealand	New Zealand Dollar	NZ$40	Lotus Development New Zealand Ltd Customer Service Dept Level 20, ASB Bank Centre Cnr Albert & Wellesley Sts Auckland New Zealand
Norway	Norwegian Krone	Nkr 190	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Portugal	Escudo	4500 Esc.	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
South Africa	South African Rand	R135	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Spain	Peseta	3500Pts	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Sweden	Krona	200 Skr	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland
Switzerland	Swiss Franc	SFr 35	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17

			Ireland
United States	US Dollar	$10	Lotus Development Corporation SmartSuite Documentation P.O. Box 25367 Rochester NY 14625-0367 USA
United Kingdom	Sterling	£15	Product Fulfillment Programme Lotus Development, Unit 12 Airways Industrial Estate Cloghran Dublin 17 Ireland

LotusScript Documentation as Online Books

If you have purchased the CD-ROM version SmartSuite, you can use SmartSuite Install to install the following Online Books about LotusScript:

- *LotusScript Language Reference*
- *LotusScript Programmer's Guide*

For more information about installing Online Books, see your SmartSuite installation documentation.

LotusScript Documentation on the Web

You can view updated versions of LotusScript documentation, or download updated sample applications or Help files from the SmartSuite Developers home page.

Enter the following URL in the location field in your browser and press ENTER:

`www.lotus.com/home.nsf/welcome/suitedev`

Overview: Designing SmartSuite Applications

LotusScript provides a variety of tools and services to support you in developing applications for SmartSuite. Getting productive in a new programming environment often involves understanding how all the pieces work together -- the tools, the language conventions, the object dependencies, and so on. Understanding how to approach the problem and where to enter your script code is half the challenge in learning.

Choosing a place to begin

Lotus Notes, 1-2-3, Approach, Freelance Graphics, and Word Pro all use the same underlying LotusScript language. Each product implements LotusObjects on top of the LotusScript language. To determine which product best supports the goals for your script application, consider using each of the SmartSuite products and reviewing its features. Read *Developing SmartSuite Applications Using LotusScript* for overviews of what each product can bring to your programming effort. Implement a couple of simple procedures in each of the products to get a feel for its features and objects. In the long run, you'll be better able to determine which product provides strengths where you need them most and how you can develop cross-product applications that take advantage of the strengths of each product.

Working the basics

LotusScript applications share the following common features.

- You need a Lotus product to run script applications.
- You need a Lotus product to store scripts in a product document such as a 1-2-3 workbook or Word Pro document.
- You need to run the Lotus Integrated Development Environment (IDE) to edit and debug scripts stored in a product document.
- You need to open an IDE window for each product document containing scripts that you want to modify.

To write a basic script application, therefore, you must run a Lotus product and load a document in that product. You can then write scripts for the product objects that you have created in your product.

Writing scripts in the Integrated Development Environment (IDE)

Your primary tool for developing script applications is the Lotus Integrated Development Environment (IDE). Beyond providing the basic tools such as an editor, a debugger, a browser, and a dialog editor, the IDE provides a high degree of integration with each Lotus product. It is easy to move between tasks that you perform in a product and those that you perform in the IDE.

Writing global scripts

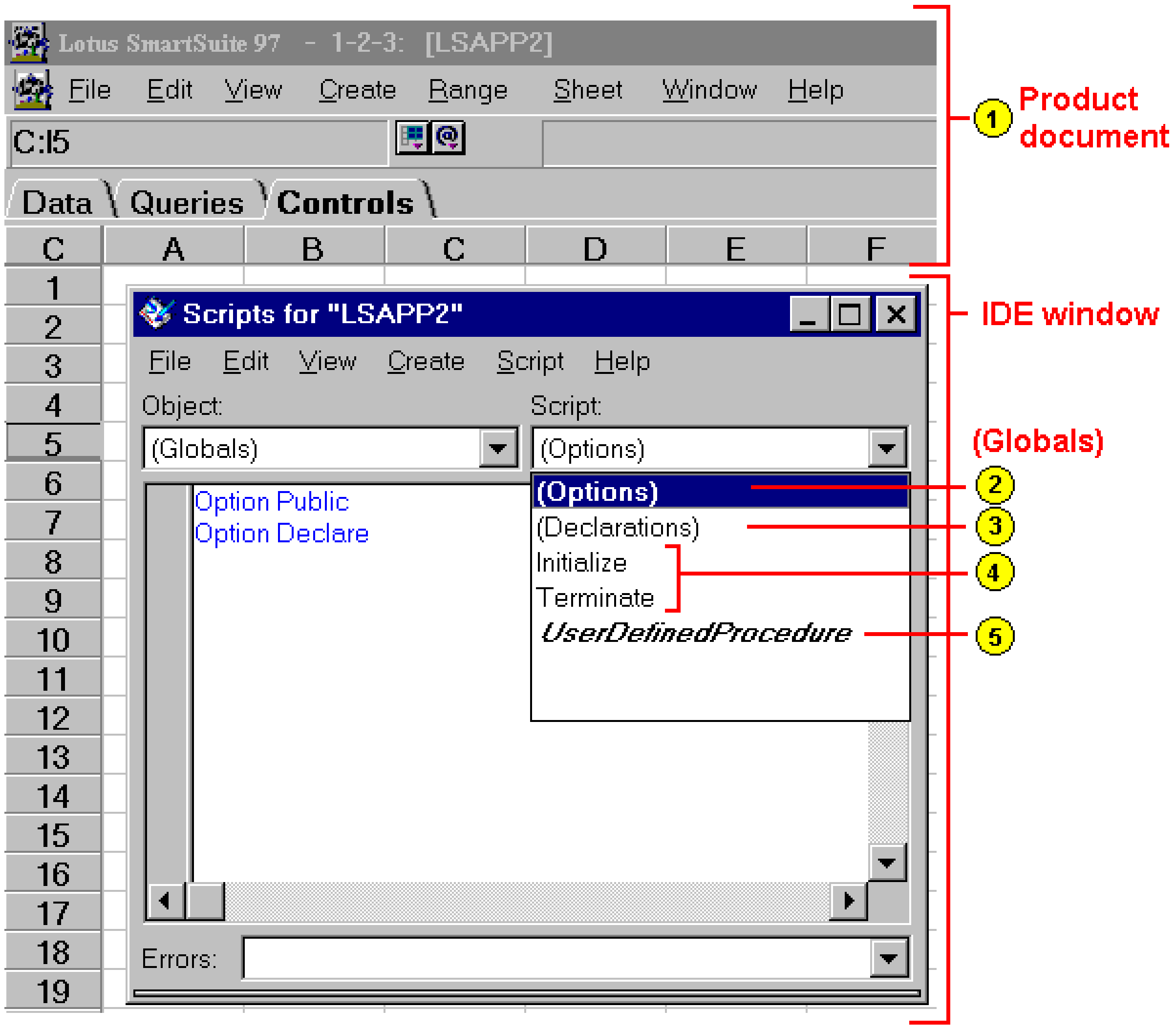
Global scripts make declarations, options, and procedures available to all scripts in your document. For example, to write global scripts for a 1-2-3 document named LSAPP2.123, you must first run 1-2-3, load the document LSAPP2.123, and then open an IDE window for that document. Choose Edit - Scripts & Macros - Show Script Editor in the 1-2-3 menu to activate an IDE window for your current document.

The IDE lists objects that you can script in the Object list and scripts for each of those objects in the Script list. You can add statements to predefined scripts in (Globals) such as (Options), (Declarations), Initialize, or Terminate or you can create your own named procedures. You do not need to modify predefined scripts to write a basic script application.

The following illustration shows how to select a particular script for (Globals).

Click any item in the following list to learn more about it.

1 Product document
2 (Options) scripts
3 (Declarations) scripts
4 Initialize and Terminate subs
5 User-defined procedures



Writing scripts for product objects

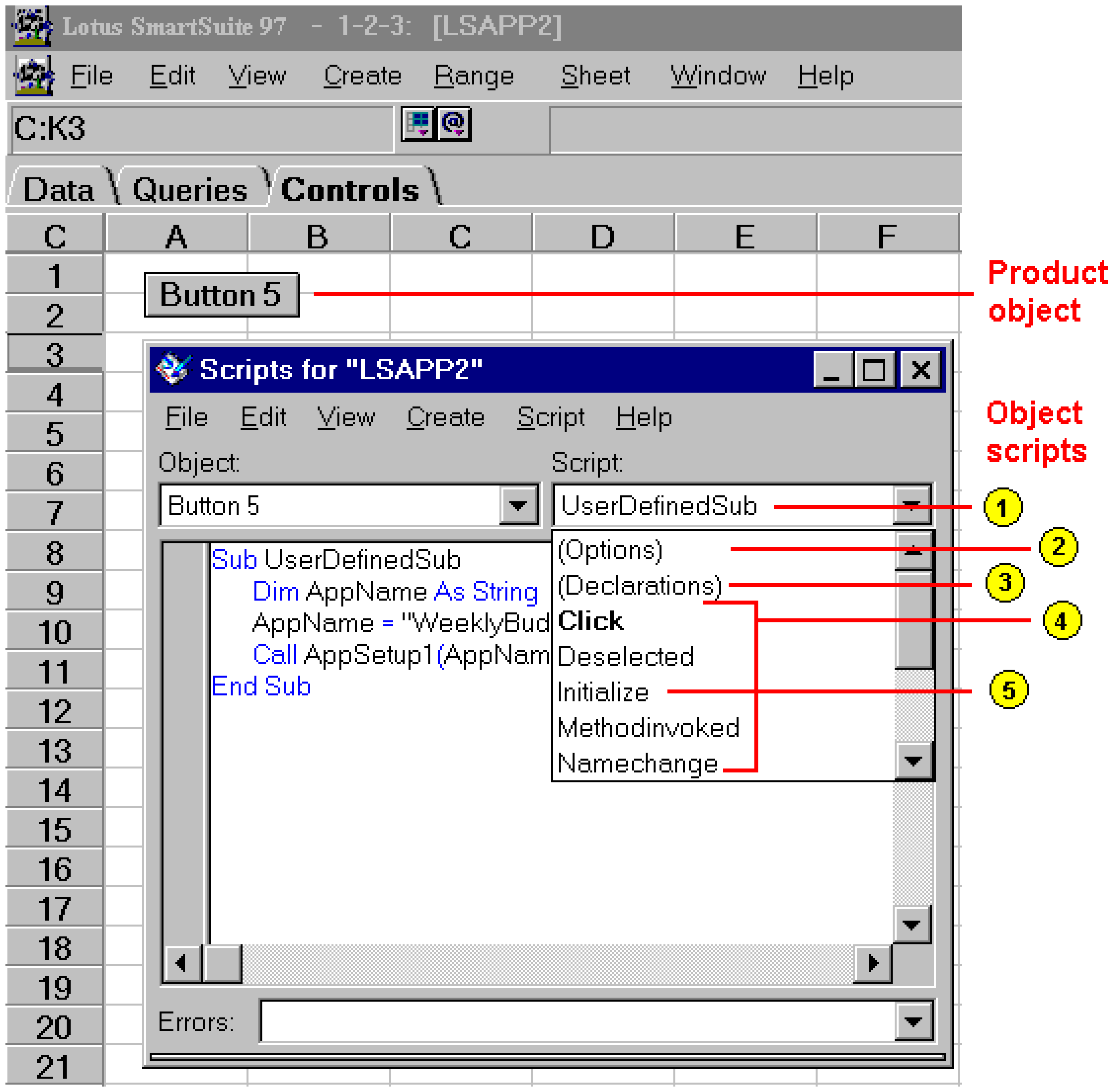
You can also write scripts for product objects in your document. As with (Globals), you can add statements in the predefined scripts for an object or create new procedures for that object. Unlike scripts that you write in (Globals), the declarations, options statements, and procedures that you write for a product object are not generally available to scripts attached to a different product object.

The predefined scripts for product objects include object event procedures. Script statements in an object event procedure are executed when an object such as a button receives a particular event in your product such as its being clicked, double-clicked, or moved. For example, if you have added a button named Button 5 to the 1-2-3 document LSAPP2.123 and you want it to run some script when you click it, you must add script statements to the Click procedure for Button 5. To select this event procedure, choose the Button 5 object in the IDE Object list and choose Click in the Script list.

The following illustration shows how to select a predefined or user-defined script for a 1-2-3 product object named Button 5.

Click any item in the following list to learn more about it.

1 User-defined procedures
2 (Options) scripts
3 (Declarations) scripts
4 Event procedures
5 Initialize and Terminate subs



```
Sub UserDefinedSub
    Dim AppName As String
    AppName = "WeeklyBud
    Call AppSetup1(AppNam
End Sub
```

Working with external script files

In many cases, the one-application-per-document approach is sufficient for working with objects and data in isolated documents. To develop more sophisticated applications that reuse important scripts or use multiple products, you should consider using the following types of external script files:

LotusScript Script (LSS) files

LotusScript Object (LSO) files

LotusScript Extension (LSX) files

OLE Custom Control (OCX) files

Dynamic-link Library (DLL) files

Dynamic-link Library (DLL) files

If you have developed useful functions in C and compiled them in a Dynamic-link Library (DLL), you can call them from your LotusScript application. For example, the following procedure declares and calls a LotusScript function named SendDLL corresponding to a C function named _SendExportedRoutine in the DLL file named MYEXPORTS.DLL.

```
Declare Function SendDLL Lib _
   "C:\LOTUS\ADDINS\MYEXPORTS.DLL" _
   Alias "_SendExportedRoutine" (i1 As Long, i2 As Long)
SendDLL(5, 10)
```

For more information on using Dynamic-link Libraries, see *LotusScript Language Reference*.

(Declarations) scripts in (Globals)

The (Declarations) script is designed to contain the following statements:

- Dim statements for variables that you want to be available to all scripts in your document
- Public, Private, Type, Class, and Declare Lib statements (external C calls)
- Const statements for those constants that you want to be available to all scripts in your document and are not needed for Use or UseLSX statements in (Options)

By default the (Declarations) script is initially empty.

If you enter Type, Class, or Declare Lib statements in any other script in (Globals), the IDE moves them to (Declarations) automatically. If you enter Dim, Public, Private, or Const statements outside the scope of a procedure in another script, the IDE moves them to (Declarations) automatically. Const statements in (Options) are the exception to this rule.

Initialize and Terminate subs in (Globals)

Initialize script
Use the Initialize sub in (Globals) to initialize variables that you have declared in (Declarations). The Initialize sub executes before any of these variables are accessed and before any other scripts in (Globals) are executed. By default, the Initialize script is empty.

Terminate script
Use the Terminate sub in (Globals) to clean up variables that you have declared in (Declarations) when you close your document or when you modify a script and execute it again. For example, you might use an Open statement to open a file containing data in Initialize and use a Close statement in Terminate to close it. By default, the Terminate script is empty.

(Options) scripts in (Globals)

The (Options) script in (Globals) is designed to contain these the following statements:

- Option statements

  Note (Options) contains the statement Option Public by default. This makes Const, Dim, Type, Class, Sub, Function, and Property statements public by default. You can use the Public form of these statements to make them public explicitly or the Private form to make them unavailable to other scripts outside (Globals).
- Def*type* statements
- Use and UseLSX statements
- Const statements needed for Use and UseLSX statements

If you enter any of these statements, except for Const, in any other script in (Globals), the IDE automatically moves them to (Options).

Option and Def*type* statements that you enter in (Options) apply only to scripts for the current object. To make certain that an option is applied consistently throughout your document, enter the appropriate statement in the (Options) script for every object for which you are writing scripts.

User-defined procedures in (Globals)

While you are working in (Globals), you can add procedures to make them available throughout your document. There are three ways to add procedures to (Globals) in the IDE:

- *Using the IDE menu*: Choose Create - New Sub or Create - New Function in the IDE menu to create new subs and functions in (Globals). The IDE automatically adds the name of the new procedure to the Script list.
- *Entering statements*: Enter a Sub, Function, or Property statement anywhere in (Globals) except within a class. The IDE automatically adds the name of the new procedure to the Script list for (Globals).
- *Importing procedures from a file*: Use File - Import Script in the IDE menu to import scripts when you are working in (Globals). These imported scripts will be available to all scripts in your document. The IDE automatically adds the name of any new procedures contained in the imported script to the Script list.

LotusScript User Assistance for SmartSuite

To help you learn how to develop LotusScript applications for SmartSuite, Lotus provides a complete library of user assistance.

The following books are available in hardcopy, Adobe Acrobat, or HTML formats in your SmartSuite package, in the *SmartSuite Application Developer's Documentation Set*, or on the World Wide Web.

Getting the Most Out of LotusScript in SmartSuite Release 9

This publication explains how SmartSuite products use the LotusScript programming language and how your business can take advantage of LotusScript in developing applications for SmartSuite.

LotusScript Language Reference

This publication provides a comprehensive summary of conventions and basic commands for the LotusScript language. *LotusScript Language Reference* provides the foundation for programming any product that supports the LotusScript programming language.

LotusScript Programmer's Guide

This publication is a general introduction to LotusScript that describes basic building blocks in the language and explains how to use them to create powerful applications.

Class Reference Help and Frequently asked Questions

Each product provides comprehensive Help on product classes, frequently asked questions about programming, and code examples. All this is delivered in an innovative Help system designed to enhance your work as a programmer.

Class reference Help and frequently asked questions are available in Help format in your SmartSuite package and in HTML format on the World Wide Web.

Example code and sample applications

Most products also provide working code to illustrate important programming techniques. You can reuse and modify this code as you develop your own applications.

Example code is available in the SmartSuite package and on the World Wide Web.

LotusScript Object (LSO) files

LotusScript Object (LSO) files contain public definitions that you can use in your script applications. If you develop a library of commonly-used declarations or procedures that you want to reuse across multiple script applications, you can collect them in a product document and use the File - Export Globals as LSO menu command to create a compiled LotusScript Object file. If this file were named WKREPORT.LSO, you would make these public definitions available to your script application by entering the following statement in the appropriate (Options) script:

```
Use "C:\LOTUS\ADDINS\WKREPORT.LSO"
```

For more information on using LotusScript Object files, see *LotusScript Language Reference*.

LotusScript Script (LSS) files

LotusScript Script (LSS) files are text files that contain LotusScript statements. You can create LSS files in any text editor. Use the %Include directive anywhere in a script to reference the contents of an LSS file. For example, to include the contents of a LotusScript Script file named STDSETUP.LSS in your application, enter the following statement:

```
%Include "C:\MYSCRIPTS\STDSETUP.LSS"
```

By default, LotusScript assumes that the LotusScript Script files that you reference have an LSS file extension. You can actually use any extension for your text file or no extension at all.

For more information on using LotusScript Script files, see *LotusScript Language Reference*.

LotusScript Extension (LSX) files

LotusScript Extension (LSX) files are Dynamic-link Libraries (DLLs) that contain public class definitions. LSX files are developed using with the Lotus LSX Toolkit. To obtain a version of the LSX Toolkit for your operating system, connect to the Lotus home page on the WorldWide Web. Lotus ships LSX files for Notes and Approach; other LSX files are being developed for SmartSuite products by Lotus and by third-party developers. These extension files expand the range of classes that you can use in your LotusScript applications.

Tip You can enter a UseLSX statement in any script; the IDE automatically moves it to (Options).

Loading and using class definitions in LSX files

There are two ways to load and use the public class definitions in an LSX file.

- If the LSX file that you want to load is not registered in the Windows Registry, you must refer to the LSX file directly in your UseLSX statement.

  ```
  UseLSX "C:\MYSCRIPTS\LSX4DB2.DLL"
  ```

- If an LSX is registered and you want to reference a class definition directly, you can enter the name of the class definition.

  ```
  UseLSX "ObjectName"
  ```

  In this example, LotusScript searches all entries under "LotusScriptExtensions" in the Windows Registry for the specified class definition and loads that definition.

  Note If the LSX file you want to load is registered in the Windows Registry, you can reference its Registry name and have Windows provide the appropriate DLL name and file path. SmartSuite registers an LSX file that contains Notes public class definitions. To use these Notes class definitions in your cross-product script applications, enter the following statement:

  ```
  UseLSX "*Notes"
  ```

Viewing class definitions

Once you have run a script containing a UseLSX statement and loaded an LSX file, you can browse its class definitions in the IDE Browser panel.

For more information on using LotusScript Extension files, see *LotusScript Language Reference*.

(Declaration) scripts in object scripts

The (Declarations) script for an object is designed to contain the following statements:

- Dim statements for variables that you want to be available to all scripts for the current object
- Const statements for those constants that you want to be available to all scripts for the current object and that are not needed for Use or UseLSX statements in (Options)

By default the (Declarations) script is initially empty.

Event procedures in object scripts

If you are writing a script for an object, the Script list displays default event procedures for the selected object. In the IDE you cannot create new event procedures for an existing product object because valid events for that object are defined by the product.

Initialize and Terminate subs in object scripts

Initialize sub

Use the Initialize sub to set up variables declared in the object's (Declarations) script. The Initialize sub for an object executes before any of its event procedures. By default, the Initialize script is empty.

Note Scripts for controls created in the Lotus Dialog Editor do not have Initialize subs.

Terminate sub

Use the Terminate sub to clean up variables that you have declared in the object's (Declarations) script. By default the Terminate script is empty.

Note Scripts for controls created in the Lotus Dialog Editor do not have Terminate subs.

(Options) scripts in object scripts

The (Options) script for an object is designed to contain these the following statements:

- Option statements
- Def*type* statements
- Use and UseLSX statements
- Const statements needed for Use and UseLSX statements

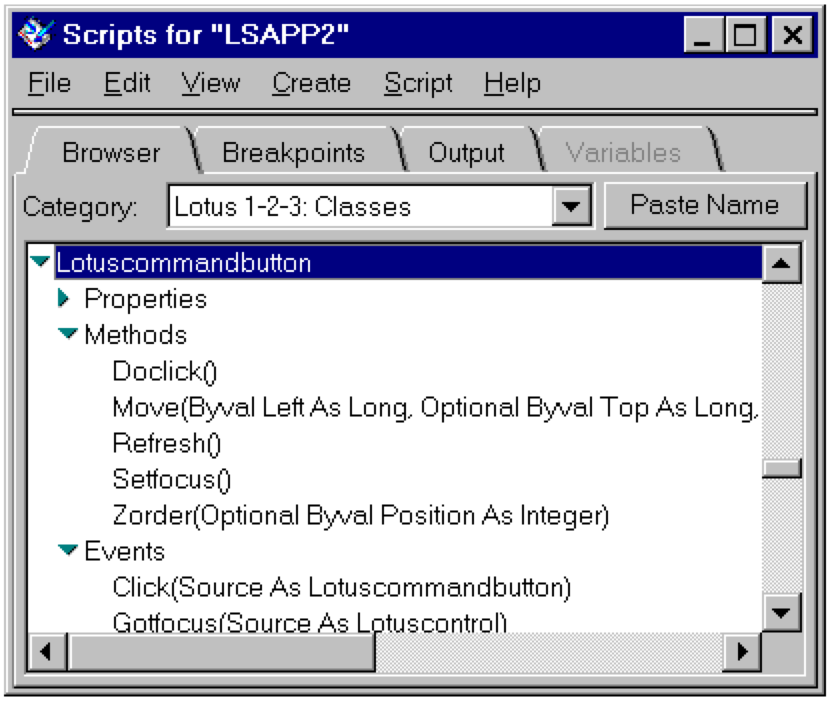
User-defined procedures in object scripts
You can create other named subs, functions, and properties for objects in addition to the predefined scripts or event procedures. Because these procedures are not in (Globals), they can be called only from other scripts for the object.

There are three ways to create object scripts in the IDE:

- *Using the IDE menu*: Use Create - New Sub and Create - New Function to create new subs and functions for an object. The IDE automatically adds the name of the new procedure to the Script list for that object.
- *Entering statements*: Enter a Sub, Function, or Property statement anywhere in a script for the current object. The IDE automatically adds the name of the new procedure to the Script list for that object.
- *Importing procedures from a file*: Use File - Import Script when you are working with object scripts to import scripts for that object. The IDE automatically adds the name of any new procedures contained in the imported script to the Script list.

OLE Custom Control (OCX) files

OLE Custom Controls extend the number of objects that you can script in Lotus products. For example, the Lotus dialog controls listed under product classes in the IDE Browser panel are OCX controls that you can add to the Lotus Dialog Editor.

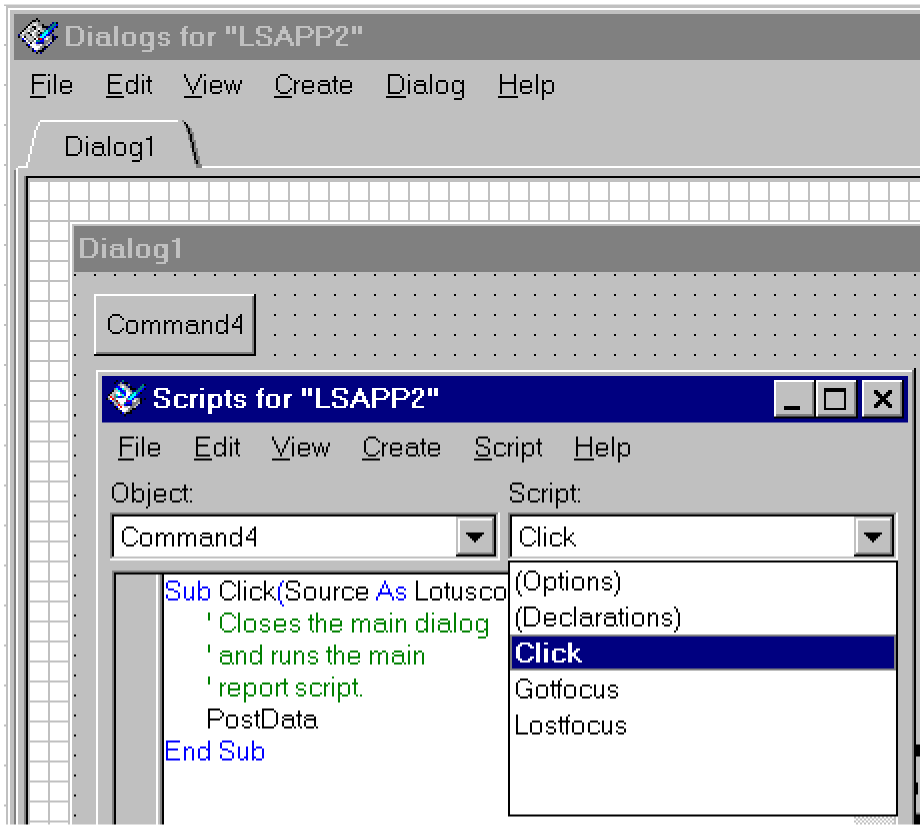


Scripts for "LSAPP2"
File Edit View Create Script Help
Browser Breakpoints Output Variables
Category: Lotus 1-2-3: Classes
Paste Name

```
Lotuscommandbutton
  Properties
  Methods
    Doclick()
    Move(Byval Left As Long, Optional Byval Top As Long,
    Refresh()
    Setfocus()
    Zorder(Optional Byval Position As Integer)
  Events
    Click(Source As Lotuscommandbutton)
    Gotfocus(Source As Lotuscontrol)
```

Once you have added an OCX control to your product, you can script its properties, methods, and events in the IDE Script Editor.

The following illustration shows how the properties, methods, and events of an Lotus CommandButton OCX named Command4 are available to you in the IDE.



```
Sub Click(Source As Lotusco
    ' Closes the main dialog
    ' and runs the main
    ' report script.
    PostData
End Sub
```

Tip You can add OCX controls registered on your system to the Lotus Dialog Editor Toolbox by choosing File - Toolbox Setup in the Lotus Dialog Editor menu.

Product Document
To edit scripts in the IDE or to execute them in one or more products, you must create or use a document in your product that contains the scripts. Lotus products supporting LotusScript use the following document extensions:

Lotus Product	*Document extension(s)*
1-2-3	123
Approach	APR
Freelance Graphics	SMC or PRZ
Notes	NSF
Word Pro	LWP

Using LotusScript Examples

Code examples provide working models for the scripts that you write. Whether the example is listed in a Help example or available as a product document on disk, you can copy statements or entire scripts from the examples and use them in your own script applications.

There are two types of LotusScript examples, each designed to illustrate a different aspect of the LotusScript language or the classes available for each SmartSuite product.

Examples in reference Help

Most examples appear in reference Help for the LotusScript language and for product classes. These brief examples focus on individual elements in the language or members of a product class. They illustrate how to use correct syntax for a working example, how to enter appropriate values for parameters, and how dependencies between elements operate.

Note Although you can copy examples from reference Help and paste them into your scripts, they are not designed primarily to be self-contained. Sometimes there are dependencies between a piece of example code and the larger sample application from which it is derived.

Examples in Frequently-asked Questions (FAQs) Help

Frequently-asked questions (FAQs) illustrate how to complete common programming tasks using LotusScript. Examples in FAQs not only illustrate how individual statements work, but they also illustrate how these statements form a complete application or procedure. Most examples in FAQs are designed to be self-sufficient; you can copy one or more procedures from Help, paste them into your own scripts in the Script Editor, and execute them.

Note When there are dependencies in an example that would require you to modify the example to make it run, these dependencies are documented in the Help topic or at the beginning of the first script in the example.

Using LotusScript Help

The design for LotusScript Help supports three of the most frequent activities that you perform as a programmer:

- Searching for objects and elements to use in your scripts
- Writing scripts
- Debugging scripts

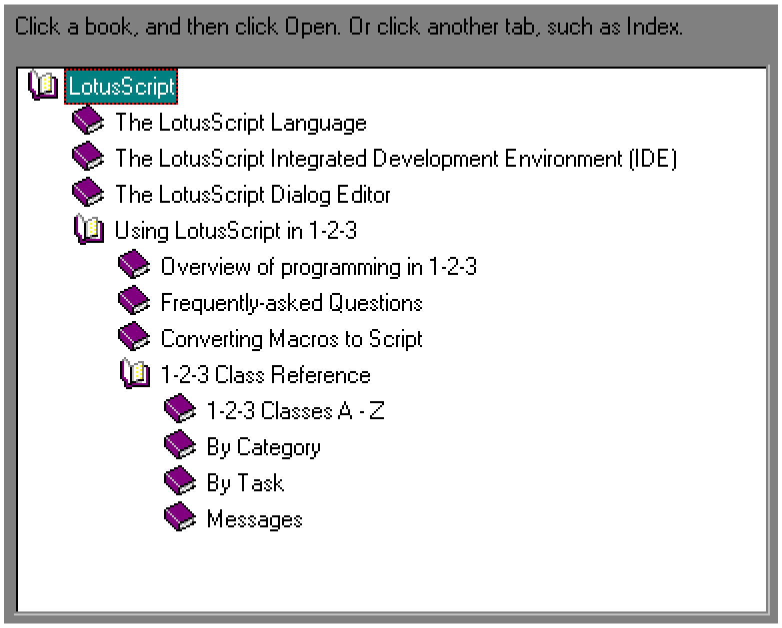
LotusScript Help uses different types of windows to display different types of information, so it is important to know what each type of window contains and how to navigate between them.

Using Help to search for objects and elements

There are areas in Help designed to help you search for objects and language elements to use in your scripts:

LotusScript Help Contents

You can use Contents in Help to examine the overall structure of Help and to browse for Help topics relevant to your current script.

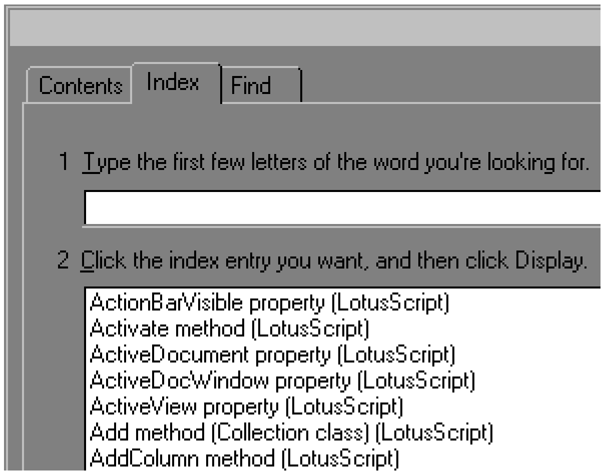


Click a book, and then click Open. Or click another tab, such as Index.

- LotusScript
  - The LotusScript Language
  - The LotusScript Integrated Development Environment (IDE)
  - The LotusScript Dialog Editor
  - Using LotusScript in 1-2-3
    - Overview of programming in 1-2-3
    - Frequently-asked Questions
    - Converting Macros to Script
    - 1-2-3 Class Reference
      - 1-2-3 Classes A - Z
      - By Category
      - By Task
      - Messages

LotusScript Index

Indexes are one of the most popular ways that programmers search for information. Topics in LotusScript Help are indexed alphabetically so you can enter key phrases or keywords and navigate to the corresponding Help topics.



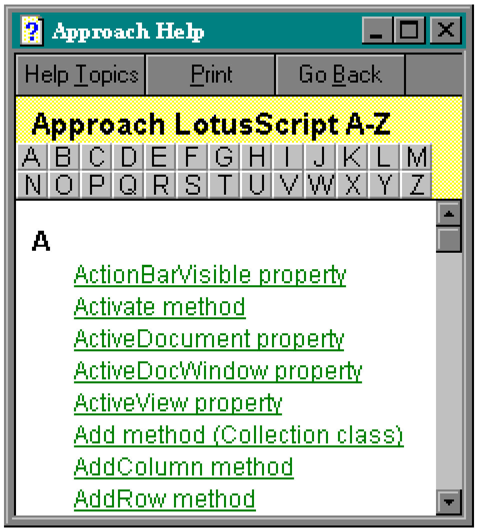
1 Type the first few letters of the word you're looking for.

2 Click the index entry you want, and then click Display.

ActionBarVisible property (LotusScript)
Activate method (LotusScript)
ActiveDocument property (LotusScript)
ActiveDocWindow property (LotusScript)
ActiveView property (LotusScript)
Add method (Collection class) (LotusScript)
AddColumn method (LotusScript)

LotusScript A - Z lists

LotusScript Help for each product provides A - Z lists of its classes, properties, methods, and events, including a comprehensive list of all the elements in the product.



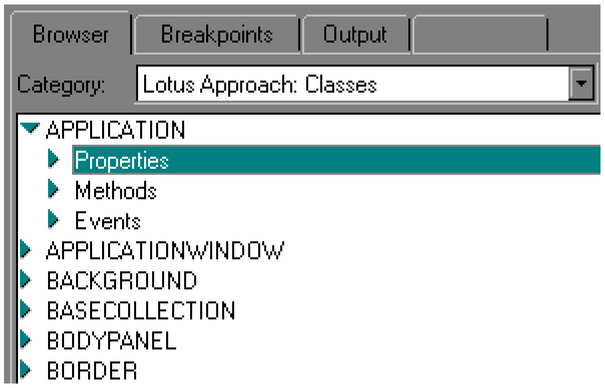
Approach LotusScript A-Z

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

A

ActionBarVisible property
Activate method
ActiveDocument property
ActiveDocWindow property
ActiveView property
Add method (Collection class)
AddColumn method
AddRow method

IDE Browser Help
The Browser panel in the Integrated Development Environment (IDE) displays lists of LotusScript language elements and classes for products. You can expand and collapse entries in the Browser to view the associated properties, methods, and events for objects.



Category: Lotus Approach: Classes

APPLICATION
Properties
Methods
Events
APPLICATIONWINDOW
BACKGROUND
BASECOLLECTION
BODYPANEL
BORDER

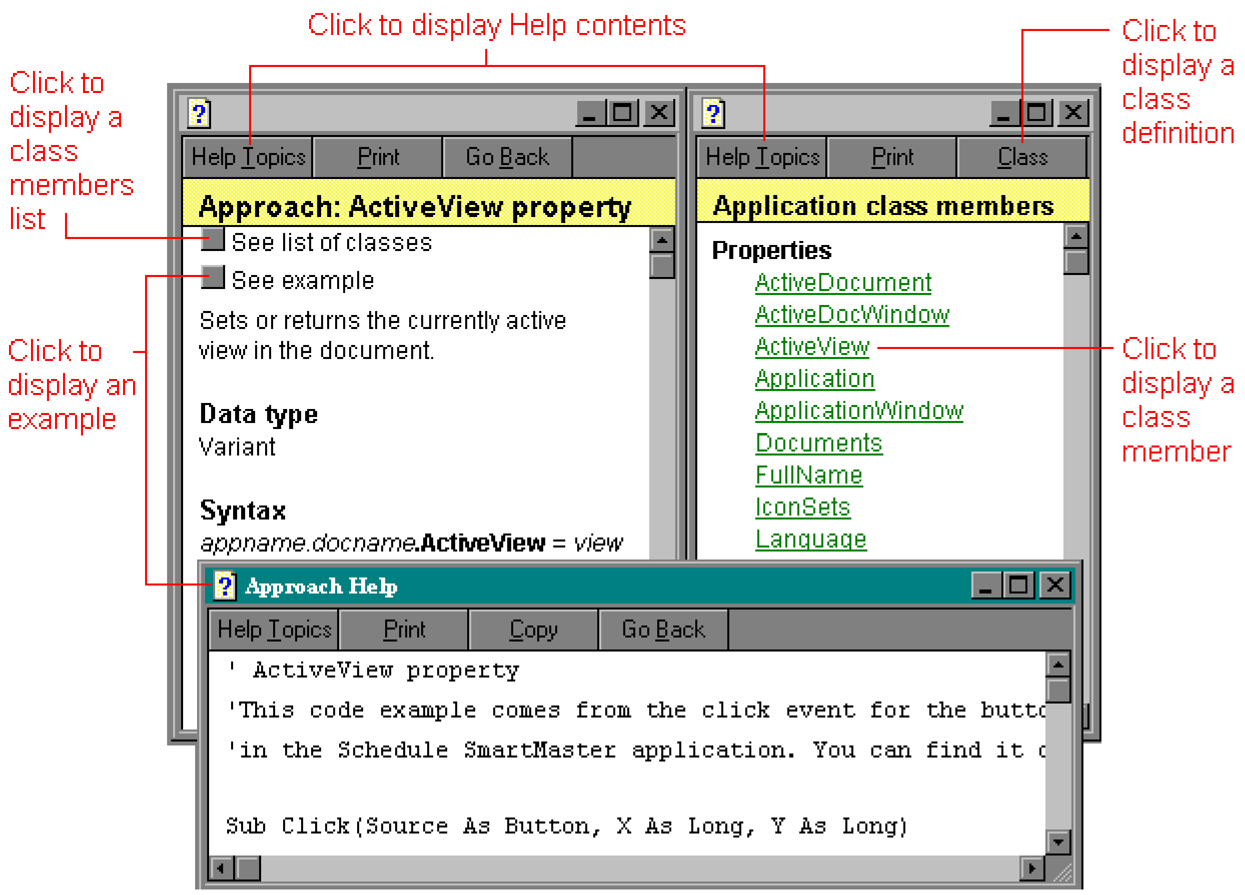
Highlight an element in the Browser panel and press F1 (HELP) to get context-sensitive Help on that element.

Using Help to write scripts
Help focuses on objects. As you are writing scripts, you explore the relationships between product classes and the behaviors of objects in that product.

Types of Help windows
To support this exploration, Help separates information about classes into four types of windows:

- Class definition windows define what a class does in a product and how it works in the product's containment hierarchy. The class definition topic for the 1-2-3 Range object describes what ranges do in 1-2-3, how they are contained by larger objects, and how they contain smaller objects.
- Class member list windows list all the properties, methods, and events that are members of a particular class.
- Class member windows focus on particular properties, methods, or events.
- Example windows contain one or more scripts for a particular property, method, or event. You can copy and paste script statements from these example windows into the IDE Script Editor.

Displaying Help windows
To display different types of LotusScript Help windows, use buttons in Help topics and in the Help window that are labeled by the type of Help window. The following illustration shows how to use buttons to display class member, class member list, and example windows in Help.


Click to display Help contents
Click to display a class members list
Click to display an example
Click to display a class definition
Click to display a class member
Help Topics
Print
Go Back
Class
Copy
Approach Help

Approach: ActiveView property

See list of classes

See example

Sets or returns the currently active view in the document.

Data type

Variant

Syntax

appname.docname.**ActiveView** = *view*

Application class members

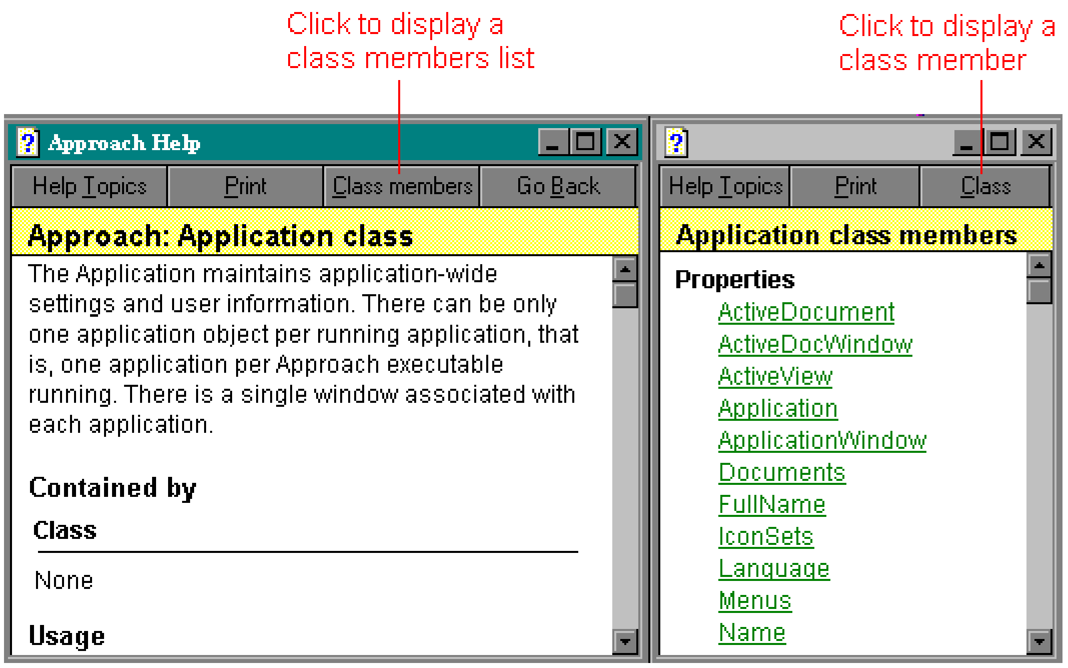
Properties

ActiveDocument
ActiveDocWindow
ActiveView
Application
ApplicationWindow
Documents
FullName
IconSets
Language

```
' ActiveView property
'This code example comes from the click event for the butto
'in the Schedule SmartMaster application. You can find it o

Sub Click(Source As Button, X As Long, Y As Long)
```

The following illustration shows how to display class definition and class member list windows in Help.


Click to display a class members list
Click to display a class member
Approach Help
Help Topics
Print
Class members
Go Back
Class

Approach: Application class

The Application maintains application-wide settings and user information. There can be only one application object per running application, that is, one application per Approach executable running. There is a single window associated with each application.

Contained by

Class
None

Usage

Application class members

Properties

ActiveDocument
ActiveDocWindow
ActiveView
Application
ApplicationWindow
Documents
FullName
IconSets
Language
Menus
Name

Help for editing and debugging scripts

You can also get context-sensitive Help about keywords and messages when you are editing or debugging your scripts in the IDE.

Context-sensitive Help in the Script Editor and Script Debugger
If you need help on a keyword while you are writing or debugging a script in the Script Editor and Script Debugger, place the insertion point on the keyword and press F1 (HELP) to get context-sensitive Help on that keyword.

Context-sensitive Help on messages
You can also get context-sensitive Help on two types of messages in the IDE. In the Script Editor, you can get context-sensitive Help on syntax errors. Navigate to the statement that caused the error and press F1 (HELP). When you are debugging your scripts and the IDE reports a run-time error, press F1 (HELP) to display information about that error and suggestions about fixing it.