Help is not available

The Help topic you selected is not available. Click Go Back to see the previously displayed Help topic, or click Help Topics and then select a different Help topic.

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.

Macro Help is not available

The topic you selected is in a Help file that is not installed by the Install program.

If you have the 1-2-3 97 or SmartSuite 97 CD-ROM, you can install macro Help from there. If not, you can download macro Help from the World Wide Web or order a copy from Lotus Customer Support. For more information, see "Installing Help on macro commands."

How do I call DLL functions?

When you create LotusScript applications for 1-2-3, you are not limited to calling LotusScript procedures. Your LotusScript applications can call any procedures that are compiled in a dynamic-link library (DLL).

To call procedures in a DLL, you need to know the following:

- The name of the DLL
- The full path for the DLL (if it is not in your default path)
- The names and parameters for procedures that you want to call

The following example calls a Win32 API function named sndPlaySound that is stored in the DLL file C:\WINDOWS\SYSTEM\WINMM.DLL.This function plays a Windows .WAV file. To use this function in a LotusScript application, first declare the function and then call it from a script.

Note Enter the following statements in (Declarations) for (Globals) if you want to call .WAV files from any script in your application.

```
' Runtime Dependencies:
' Files and paths: WINMM.DLL must be installed in C:\WINDOWS\SYSTEM
' or somewhere in your current path. The sound file
' OFF2RACE.WAV must be installed in the subdirectory
' C:\WINDOWS\MEDIA.
' Declare a return value to use when you call the DLL
' function in a script.
Dim SoundReturnValue As Integer
' Declare the DLL function as a public function in LotusScript.
Declare Public Function sndPlaySound Lib "winmm"_
   Alias "sndPlaySoundA" _
   ( Byval WaveFile As String, Byval theFlags As Long ) _
   As Integer

' Declare some of the constants used by parameters of the DLL function.
Public Const SND_SYNC            = &H0000  ' play synchronously (default)
Public Const SND_ASYNC           = &H0001  ' play asynchronously
Public Const SND_NODEFAULT       = &H0002  ' silence (!default) if sound not found
Public Const SND_MEMORY          = &H0004  ' pszSound points to a memory file
Public Const SND_LOOP            = &H0008  ' loop the sound until next sndPlaySound
Public Const SND_NOSTOP          = &H0010  ' don't stop any currently playing sound
```

The following script calls the declared function and specifies a .WAV file to play.

```
Sub TestSoundFiles
   SoundReturnValue = sndPlaySound( "C:\WINDOWS\MEDIA\OFF2RACE.WAV", SND_SYNC )
End Sub
```

How do I create a custom menu?

You can use LotusScript to customize 1-2-3 menus or to create your own menus and display them in the 1-2-3 menu bar. You assign LotusScript procedures to your custom menu items. When users select a custom menu item, the associated script runs. These examples show some of the ways you can use LotusScript to customize menus in 1-2-3.

Adding a command to a pull-down menu

This example adds the command Payroll to the beginning of the 1-2-3 File menu. When the user chooses the Payroll command, 1-2-3 runs the sub OpenPayroll.

```
Sub AddPayroll
   Dim FileMenu As Menu
   Set FileMenu = CurrentApplication.CurrentMenuBar.GetMenu(1)
   FileMenu.AddItem 0, "&Payroll", "Open the Payroll template", ThisDocument,
   "OpenPayroll"
End Sub
Sub OpenPayroll
   MessageBox "You selected the Payroll command."
End Sub
```

Removing a command from a pull-down menu

This example removes the Payroll command that the previous example added to the 1-2-3 File menu.

```
Sub RemovePayroll
   Dim FileMenu As Menu
   Set FileMenu = CurrentApplication.CurrentMenuBar.GetMenu(1)
   If FileMenu.GetItemText(1) = "&Payroll" Then
      FileMenu.RemoveItem(1)
   End If
End Sub
```

Adding a pull-down menu

This example adds the menu Functions to the 1-2-3 main menu, after the Help menu.

```
Sub SetMenu
   'Set Constants for menu and menu items.
   Const M_Functions ="F&unctions"
   Const M_IsOdd ="Is&Odd"
   Const M_IsEven ="Is&Even"
   ' Set Constants for menu item descriptions.
   Const P_Functions ="Select a custom @function"
   Const P_IsOdd = "Returns True for an odd value"
   Const P_IsEven ="Returns True for an even value"
   ' Set Constants for global function names
   Const S_IsOdd ="PutIsOdd"
   Const S_IsEven ="PutIsEven"
   Dim FuncMenu As Menu
   Dim MainMenu As MenuBar
   Set MainMenu = CurrentApplication.CurrentMenuBar
   Set FuncMenu = CurrentApplication.NewMenu
   FuncMenu.MenuText = M_Functions
   FuncMenu.MenuPrompt = P_Functions
   'This statement prevents duplicate menus if the Functions menu is already in the
   menu bar.
   If MainMenu.GetItemText(-1) = "F&unctions" Then
      MainMenu.RemoveItem(-1)
   End If
   ' Add FuncMenu to the end of the 1-2-3 Main menu
```

```
   Call CurrentApplication.CurrentMenubar.AddMenu(-1, FuncMenu)
   'Add the two menu items to FuncMenu
   Call FuncMenu.AddItem(-1, M_IsOdd, P_IsOdd, ThisDocument, S_IsOdd)
   Call FuncMenu.AddItem(-1, M_IsEven, P_IsEven, ThisDocument, S_IsEven)
End Sub
Sub PutIsOdd
   'Enter the IsOdd function in the current cell
   'and leave the product in Edit mode.
   .UpdateSheetDisplay = False
   Selection.Contents = "@ISODD(x)"
   Sendkeys "{F2}"
   .UpdateSheetDisplay = True
End SubEnd Sub
Sub PutIsEven
   'Enter the IsEven function in the current cell
   'and leave the product in Edit mode.
   .UpdateSheetDisplay = False
   Selection.Contents = "@ISEVEN(x)"
   Sendkeys "{F2}"
   .UpdateSheetDisplay = True
End Sub
'This function returns 1 for an odd number; 0 for any other value.
Function ISODD (x As Integer) As Integer
   If x Mod 2 <> 0 Then
      ISODD  = 1
   Else
      ISODD = 0
   End If
End Function
'This function returns 1 for an even number; 0 for any other value.
Function ISEVEN (x As Integer) As Integer
   If x Mod 2 <> 0 Then
      ISEVEN  = 0
   Else
      ISEVEN = 1
   End If
End Function
```

Removing a pull-down menu

This example removes the Functions command that the previous example added to the 1-2-3 main menu.

```
Sub ResetMenu
   Dim MainMenu As MenuBar
   Set MainMenu = CurrentApplication.CurrentMenuBar
   If MainMenu.GetItemText(-1) = "F&unctions" Then
      MainMenu.RemoveItem(-1)
   End If
End Sub
```

Disabling a menu item

This example disables the 1-2-3 Create - Object command. When a command is disabled, it is dimmed in the menu.

```
Sub GreyCreateObject
   Dim CreateMenu As Menu
   Set CreateMenu = CurrentApplication.CurrentMenuBar.GetMenu(4)
   CreateMenu.DisableItem (-1)
```

```
End Sub
```

Enabling a menu item

This example enables the 1-2-3 Create - Object command after it was previously disabled.

```
Sub EnableCreateObject
   Dim CreateMenu As Menu
   Set CreateMenu = CurrentApplication.CurrentMenuBar.GetMenu(4)
   CreateMenu.EnableItem (-1)
End Sub
```

Creating keyboard shortcuts

When you enter command names, & (ampersand) followed by a character creates a keyboard shortcut for a command. The letter that follows the & (ampersand) appears underlined; the user can choose this command from the keyboard by pressing ALT plus the underlined letter. For example, if you enter First&Quarter, 1-2-3 displays FirstQuarter; the user can press ALT+Q to select the command. To display & (ampersand) in the command name, enter && (two ampersands). For example, to display B&W, enter B&&W.

{button ,AL(`;H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See related topics

How do I call other Windows applications?

Use the LotusScript Shell command to call operating system services or utilities. The following example starts the Microsoft Windows Help program and displays the 1-2-3 Help topic "Details: Recording a script."

```
Sub Click(Source As Button, X As Long, Y As Long, Flags As Long)
' Runtime dependencies
' Files: This example uses a help file and a help topic
' that was installed with 1-2-3
' SS1N60EN.HLP is the 1-2-3 main Help file.
' H_123_RECORDING_A_SCRIPT_DETAILS is
' the Help context ID of a topic in that file.
' Substitute the name of another help file and context ID
' to call your own Help topic.
' Platform: This script calls a Windows-specific service, WinHelp.
   ' Declare a variable to store the return value of the call to WinHelp.
   Dim HelpReturnValue As Integer
   ' Use the LotusScript Shell command to call WinHelp.
   ' The command line switch -I specifies a Help context ID.
   HelpReturnValue = Shell("WINHLP32.EXE -I H_123_RECORDING_A_SCRIPT_DETAILS _
      SS1N60EN.HLP", 1)
End Sub
```

How do I display dialog boxes?

You can create and display custom dialog boxes, as well as to display built-in LotusScript and 1-2-3 dialog boxes.

Creating a custom dialog box

To open the Dialog Editor in 1-2-3, select Edit - Scripts & Macros - Show Dialog Editor. 1-2-3 saves dialog boxes you create in the Dialog Editor with the .123 file in which you created them.

Displaying a custom dialog box

You can use the LotusScript Dialog Editor to create and edit custom dialog boxes, and you can use the Show method to display these dialog boxes in 1-2-3.

The following example displays a custom dialog box named EmpInfoDlg:

```
Sub GetInfo
   EmpInfoDlg.Show
End Sub
```

Displaying built-in 1-2-3 dialog boxes and InfoBoxes

You can use the {Dialog?} macro command to display a number of 1-2-3 dialog boxes and InfoBoxes. The following table lists the dialog boxes and InfoBoxes you can display and the {Dialog?} command you use to display them.

Dialog box	{Dialog?} command
File - New Workbook	{Dialog? "File-New"}
File - Open	{Dialog? "Open"}
File - TeamMail	{Dialog? "Send-Mail"}
File - Save As	{Dialog? "Save-As"}
File - Save Copy As	{Dialog? "Save-Copy-As"}
File - Workbook Properties	{Dialog? "Doc-Info"}
File - Print	{Dialog? "Print"}
File - Preview & Setup - Printer	{Dialog? "Printer-Setup"}
File - User Setup - 1-2-3 Preferences	{Dialog? "User-Setup"}
Edit - Clear	{Dialog? "Clear"}
Edit - Paste Special	{Dialog? "Paste-Special"}
Edit - Go To	{Dialog? "Go-To"}
Edit - Find and Replace	{Dialog? "Find-and-Replace"}
Edit - Check Spelling	{Dialog? "Spell-Check"}
View - Set View Preferences	{Dialog? "Set-View-Preferences"}
Range - Name	{Dialog? "Range-Name"}
Range - Sort	{Dialog? "Range - Sort"}
Range - Parse	{Dialog? "Parse"}

Use the MacroRun or MacroRunText method to run a macro as part of a script.

Displaying built-in LotusScript dialog boxes

You might not always have time to create your own dialog boxes. When you only want to display a message or result, use the LotusScript MessageBox function. When you only want to get simple data from users, use the LotusScript InputBox function.

The following code displays a dialog box that prompts the user to enter an ID number and then converts the user's input from a string to an integer:

```
Dim num As Integer
num% = CInt(InputBox$("Enter your ID number:","Login"))
```

The following code displays a message if the cell named COST is blank:

```
If [COST].Contents = "" Then
   MessageBox "You must enter a cost."
Else
   NextSub
```

How do I freeze the screen while a script is running?

Sheet updates that occur while a script is running can be distracting to users. The following code suppresses the display of sheet updates while a script is running:

```
CurrentApplication.UpdateSheetDisplay = False
```

The following code restores normal display of sheet updating:

```
CurrentApplication.UpdateSheetDisplay = True
```

How do I hide and display parts of the 1-2-3 user interface?

You can write a script that hides or displays specific parts of the 1-2-3 user interface, such as sets of SmartIcons.

The following example sets the display of the 1-2-3 user interface.

```
Sub SetMyView
   'Set View Preferences: Hide the sheet frame, grid lines, and sheet tabs.
   CurrentDocument.ShowSheetFrame = False
   CurrentDocument.ShowGridLines = False
   CurrentDocument.ShowSheetTabs = False
   'Set 1-2-3 Preferences: Maximize the 1-2-3 window, set the default font
   'to TimesNewRoman 10 point.
   CurrentApplication.ApplicationMaximized = True
   CurrentApplication.DefaultFontName = "TimesNewRoman"
   CurrentApplication.DefaultFontSize = 10
   'Hide the status bar and all sets of SmartIcons.
   .IconBarsVisible = False
   .StatusBarVisible = False
End Sub
```

How do I write a script for an object not listed in the IDE?

Some objects you want to write scripts for cannot be selected in 1-2-3 and do not appear in the IDE Object drop-down box. When you write scripts for these objects, you must manually create an object variable.

For example, suppose you write a script that you want to run when a particular DocWindow object gets the focus. You cannot select a DocWindow object in 1-2-3 or from the Object drop-down box. To associate the event script named MyGetFocusHandler with the event GetFocus of the first DocWindow object, you could use the following script in the Opened event script.

```
Sub Opened(source As Document)
   Dim myDW As DocWindow
   Set myDW = ThisDocument.DocWindows(0)
   On Event GetFocus From myDW Call MyGetFocusHandler
End Sub
```

You could then put your GetFocus event handler script in the (Globals) object in the IDE.

How do I reference objects in a script?

When you write scripts in 1-2-3, you can reference a 1-2-3 object by its name, provided you enclose the name in [ ] (square brackets). The following examples show several 1-2-3 objects referenced by name:

```
'Bold data in the range February
[February].Font.Bold = True
```

```
'Change the chart MyChart to a doughnut chart
[MyChart].Type = $Doughnut
```

```
'Protect all cells in the sheet Budget
[Budget].IsProtected = True
```

```
' Delete the version WorstCase for the range Q2Sales
[Q2Sales.WorstCase].DeleteVersion
```

You can also reference a range by its address enclosed in [ ] (square brackets):

```
'Add a border and border style to A:A1..A:B10
[A:A1..A:B10].OutlineBorder.Style = $SolidBorder
```

```
'Change the background color of sheet A to cornflower
[A].Background.BackColor.ColorName = "Cornflower"
```

By default, names within [ ] (square brackets) refer to objects in the current workbook. To reference objects in other workbooks, use a file reference:

```
[<<filename>>objectname]
```

For example, the following statement references a range named MyRange in the workbook D:\LOTUS\WORK\123\MYFILE:

```
[<<D:\Lotus\Work\123\Myfile>>MyRange].CopyToClipboard
```

Referencing collections

A collection is two or more ranges, selected at the same time, so that your next action affects all the ranges in the collection at once. In the rare case that you want to reference a collection, the collection must be the current selection:

```
'Create a collection and change its background color
[A:A2..A:A20].Select
[B:A2..B:A20].AddToSelection
[C:A2..C:A20].AddToSelection
Selection.Background.BackColor.ColorName = "parchment"
```

There is one exception to this rule. You can assign a print range to a collection:

```
Set .CurrentPrintSettings.PrintSelection = [A:B1..A:C25,B:B1..B:C25]
```

Type qualifiers

If multiple objects in a workbook share the same name, use a type qualifier to specify which object you want the script to act on. The syntax for a reference that includes a type qualifier is:

```
[objectname:type]
```

For example, suppose a workbook contains a map named Sales and a chart named Sales. Use the following reference for the map:

```
[Sales:Map]
```

Use the following reference for the chart:

```
[Sales:Chart]
```

If you do not include a type qualifier in a reference, 1-2-3 first looks for a range with the specified name, because range objects have the highest order of precedence in a workbook. The order of precedence for all other objects within a workbook is neither guaranteed nor consistent.

For example, if a workbook contains a range named Sales, a chart named Sales, and a map named Sales, the following statement always selects the range:

```
[Sales].Select
```

If a workbook contains only a chart named Sales and a map named Sales, unexpected results occur. The statement selects either the map or the chart, arbitrarily.

To avoid this situation, you should not assign the same name to two objects in the same workbook. However, if a workbook contains multiple objects with the same name, always include type qualifiers in your references.

For more information about naming objects in 1-2-3, search on "Naming, conventions for" in the 1-2-3 Help Index.

How do I run scripts when 1-2-3 starts or I open a workbook?

You can make your scripts run when a user starts a 1-2-3 session or opens a particular workbook. To make sure these types of scripts run when you expect them to, choose File - User Setup - 1-2-3 Preferences and select "Run file Opened scripts, autoexecute macros."

Running a script when you open a document

To run a script whenever the user opens a particular .123 file, attach the script with to the file's Opened event:

1. Choose Edit - Scripts & Macros - Show Script Editor.

   1-2-3 displays the Script Editor
2. Under Object, select the current file.
3. Under Script, select Opened.

   1-2-3 displays the empty Opened sub in the Script Editor.
4. Enter statements in the sub that you want 1-2-3 to execute when the sub runs.
5. Save the script by saving the .123 file.

When you open the file, 1-2-3 automatically runs the script.

Example

The following example adds a custom menu item to the end of the 1-2-3 Edit menu when you open the file the script is saved in:

```
Sub Opened(Source As Document)
   Dim EditMenu As Menu
   Set EditMenu = CurrentApplication.CurrentMenuBar.GetMenu(2)
   EditMenu.AddItem -1, "&Payroll", "Perform payroll calculations", ThisDocument,
   "PayCalc"
End Sub
Sub Paycalc
   MessageBox "You selected PayCalc."
End Sub
```

Running a script at the start of a 1-2-3 session

Running a script at the start of a 1-2-3 session is similar to running a script when 1-2-3 opens a workbook. Attach the script to the file's Opened event, then store the file in the "Automatically opened files" directory.

At the start of each session, 1-2-3 opens all the files in the "Automatically opened files" directory in alphabetical order, words before numbers. Any scripts attached to the Opened event of a file run when 1-2-3 opens the file.

How do I use OLE objects?

New OLE 2 automation capabilities let you develop cross-product scripts that can interact with and control products and their objects from outside. In other words, you can program other products to automate repetitive tasks.

You can use other products that support OLE automation to externally access and manipulate 1-2-3 and objects in it. For example, you can create a script in a Notes document that uses OLE automation to perform calculations in 1-2-3 and then bring the data back into Notes.

It's just as easy to use 1-2-3 to control other products using OLE automation. For example, you can use 1-2-3 to access Word Pro and Freelance Graphics, and then bring the text and graphics back into 1-2-3 where you can consolidate them with your data for a monthly report.

Object names for applications

All SmartSuite object models have an Application object. From an Application object, you can traverse the hierarchy to find all other objects. To create an OLE Automation object in a Lotus product that has LotusObjects, use your scripting language's CreateObject method. Lotus products use the following object names when exposing their objects for OLE Automation:

- Lotus123.Workbook

  Note Note Unlike the other SmartSuite products, 1-2-3 returns an object of type Document, instead of an object of type Application. To access the 1-2-3 Application object, once you have accessed the Document object, use the Parent property of the Document object (Document.Parent).
- Approach.Application
- Freelance.Application
- WordPro.Application

Embedding OLE 2 objects in your document

The following example embeds a WordPro object in a 1-2-3 document and displays the object for editing or viewing:

```
Sub EmbedNew
   'Embed a new, blank WordPro document.
   [A].NewObject 4905,1350,6015,3345,"WordPro.Document",,,False,,,
   [OLE 1].Select
   Selection.Verb $OLEVerbShow
End Sub
```

Tip The names of the OLE servers that you can specify in a NewObject statement may be slightly different from the names that appear in the "Object type" list in the Create Object dialog box. The specific names to specify in a NewObject statement are available in the Windows Registry in HKEY_CLASSES_ROOT.

Embedding files as OLE objects

The following example embeds an existing WordPro file in a 1-2-3 document. In order to make this example work, you must first create a WordPro document called C:\LOTUS\WORK\WORDPRO\MYDOC.LWP.

```
Sub EmbedFile
   'Embed a copy of a WordPro document.
   [A].NewObject 2475,2025,3855,3630,,"C:\LOTUS\WORK\WORDPRO\MYDOC.LWP",False,False,,,
   [OLE 1].Select
End Sub
```

The following example creates a link to the contents of the file, rather than creating an embedded object containing a copy of the whole file:

```
Sub EmbedFile
   'Create a link to the contents of a WordPro document.
   [A].NewObject 2475,2025,3855,3630,,"C:\LOTUS\WORK\WORDPRO\MYDOC.LWP",True,False,,,
   [OLE 1].Select
End Sub
```

class
A data type that is a description or a definition of a part of 1-2-3 that you can manipulate, as an object, in a script. For example, a Range object from the 1-2-3 Range class represents a range in a sheet.

current workbook
The workbook in which you are working. The current workbook contains either the cell pointer or a selected graphic object.

drag
To press the mouse button and hold it while moving the mouse.

Event
An action to which an application responds. The action can be performed by a user, such as a mouse click, or generated by the system, such as the elapsing of a set amount of time on the computer's clock. Each LotusObject can respond to a set of events that are predefined for the class that the object is an instance of. Events are the primary way to execute scripts. A script that is attached to an object event is executed when the event occurs. For example, a script that is attached to a document's Open event executes when the user opens the document in 1-2-3.

file reference
A file name and extension, with or without a path, enclosed in << >> (double angle brackets); for example, <<SALES.123>>@SUM(A10..B22). Use a file reference in formulas and commands to refer to data in a workbook other than the current workbook.

Function
A named procedure that performs a specific task and returns a single value (unlike a sub, which does not return a value). LotusScript provides a set of built-in functions that you can use to perform a variety of common numeric, date and time, string, data-conversion, and value-testing operations. You can also write your own functions in the Script Editor.

Functions are comprised of the following:

- The Function keyword followed by the name of the function
- A series of one or more statements that are executed as a block when you call the function
- The End Function statement which marks the end of the function's definition.

For example:

```
Function CubeNumber (intArg%)
   'Calculate the cube of intArg% and
   'make it the return value of CubeNumber
   CubeNumber = intArg% ^ 3
End Function
```

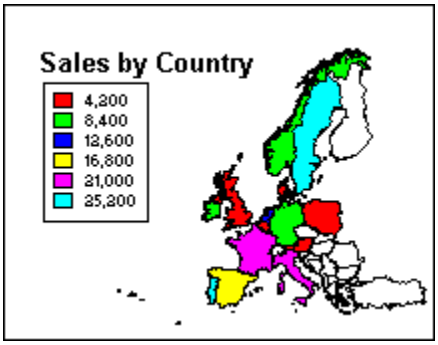
LotusScript Integrated Development Environment (IDE)
A set of tools for creating and debugging scripts in Lotus products. These tools include a script editor, a script debugger, and a dialog editor.

macro
A set of instructions, called macro commands, that automate a 1-2-3 task. You can use a macro to enter data or to perform a series of 1-2-3 commands to style sheets or workbooks, guide users through specific applications, calculate complex formulas with variable data, extract records from a database table, and so on.

map data bin

A group of values or labels in a set of map data. 1-2-3 displays each bin as a color in the map. If you have two sets of map data, 1-2-3 creates pattern bins as well as color bins.

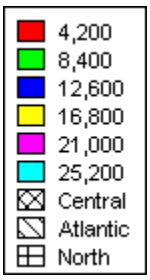
In this map, 1-2-3 groups the sales data for 15 countries into 6 bins.



The legend labels indicate the upper limit of data in that bin. For example, each country that falls into the red bin has sales less than or equal to 4,200.

map legend

Explains the meaning of the colors and patterns in a map. Values used as legend labels either exactly match values in the range of map data, or they represent the upper limit of the values contained in the bin. Use Map - Color Bins or Map - Pattern Bins to change the labels, the colors and patterns, and the values used to create bins.



Method
An action performed on or by an object. The action changes the object or gives you control over certain aspects of the object's behavior. For example, the Document object has a NewSheet method that creates a new sheet in the file.

named print styles
Print options, such as margins, headers, and footers, that you name and save with File - Preview & Page Setup (Named Style tab). When you save print styles, you can use them again.

Named print styles are stored in the workbook file. You can copy print styles among active workbooks.

Object

A component of LotusScript that you can manipulate, as you would a variable, in a script, using the properties, methods, and events associated with that object. An object is an instance of a class; the properties, methods, and events defined for that class can be applied to any object of that class.

For example, the Range object represents a range in a sheet. You might want to write a script that checks the data in a range and performs actions based on the values in that range. To do this, you use the Range object, along with its methods and properties.

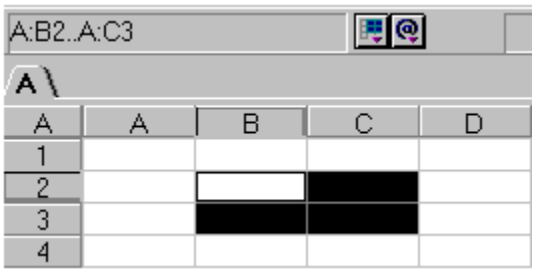
OLE (Object Linking and Embedding)
A method for linking data between applications or embedding objects created with one application into files created with another application.

Use Edit - Paste Link, Edit - Paste Special, or Create - Object to create a link or embed an object in a 1-2-3 workbook.

Property
Attributes that are associated with a class and that define the appearance and behavior of objects that are instances of that class. For example, the Font property of the Range class determines the font for the data in a Range object.

quadmillipoint, defined
A unit of distance equal to 1/256 of a point (0.0000543 inch). Used to measure draw objects. For example, rectangles.

range
A single cell, a rectangular block of adjoining cells, an entire sheet, or an entire workbook. A range is represented as the addresses of its top left and bottom right cells, separated by two periods, for example, A:B2..A:C3.



A 3D range spans two or more contiguous sheets; for example, A:B1..B:B5.

Script Debugger
A window in which you can set, clear, disable, and enable breakpoints and step through scripts to locate the source of problems that may occur while a script is running.

Script Editor
A window in which you can write and edit scripts, check script syntax, and set breakpoints for debugging scripts. The Script Editor initially displays a script associated with the selected object.

Script
A sequence of one or more LotusScript statements. A script can be a complete application or part of an application.

Sub
A named procedure that performs a specific task without returning a value (unlike a function, which does return a value).

Subs are comprised of the following:

- The Sub keyword followed by the name of the sub
- A series of one or more statements that are executed as a block
- The End Sub statement which marks the end of the sub's definition

For example:

```
Sub BlankCell ()
   [A10].Clear $Contents
End Sub
```

text block

A graphic object, shaped as a rectangle or square, that contains text. Text blocks, like other graphic objects, can be moved, copied, and sized. You can also edit, align, and change the font of the text in a text block.

Choose Create - Text to create a text block. Select the text block and use Drawing - Drawing Properties to change the appearance of a text block.

twip, defined
A unit of distance equal to 1/20 of a point (1/1440 inch). Used to measure screen objects other than draw objects. For example, window size.

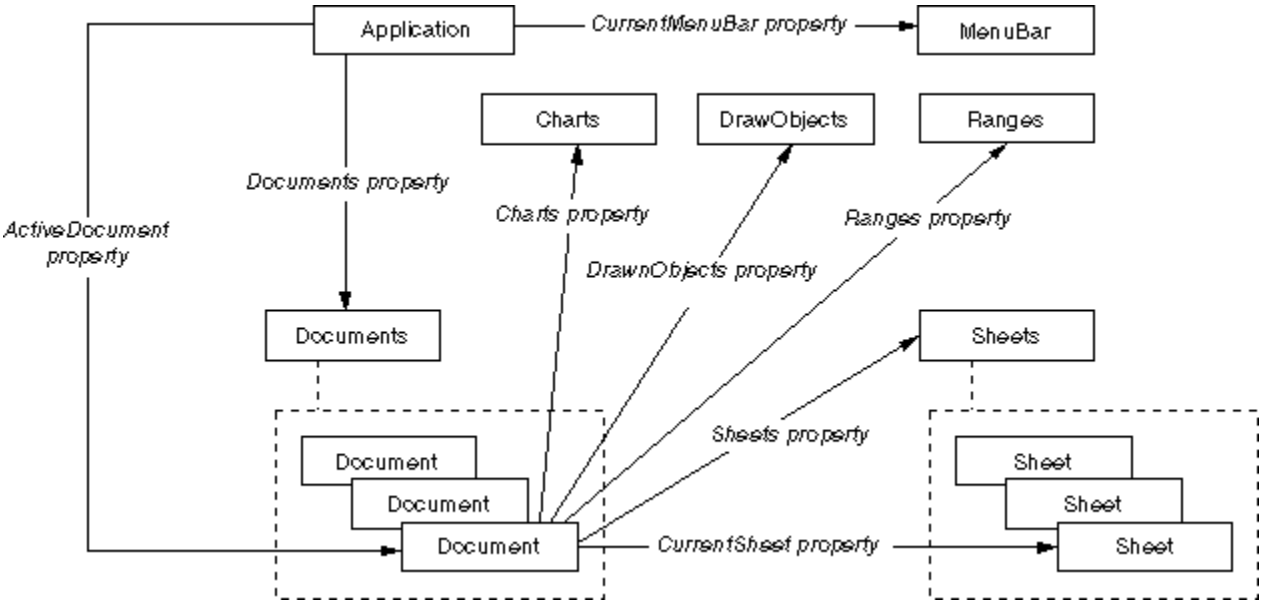
version group
A named group of one or more versions. Each version in a version group must be associated with a different named range.

In previous releases of 1-2-3, version groups were called scenarios.

version
Versions are sets of different data for the same named range. Each version has a name. 1-2-3 keeps track of the date and time the version was created or modified, and the name of the person who created or modified it. You can also assign styles to a version and attach a descriptive comment.

1-2-3 containment hierarchy

Like all SmartSuite product object models, the 1-2-3 object model is organized by containment hierarchies, which describe the containment relationships of the 1-2-3 classes.



Converting macro buttons to script buttons

When you open a .WK4 file, 1-2-3 automatically converts macro buttons that you created with previous releases of 1-2-3 into buttons that run scripts.

Macros stored in a sheet

If a button ran a macro stored in a sheet, 1-2-3 creates the following Click event script for the button:

```
Sub Click(Source As ButtonControl)
     [file.range].MacroRun
End Sub
```

file is an optional file reference. You need to specify the file only if the macro is stored in a different file than the button, for example, if the macro is stored in a macro library.

range is the name or address of the first cell in the macro.

Example

The following sub runs the macro named INTEREST stored in the active file D:\LOTUS\WORK\MYMACROS.WK4:

```
Sub Click()
     [<<d:\lotus\work\mymacros>>.interest].MacroRun
End Sub
```

Macros stored with a button

If a button ran a macro stored with the button in the Assign Button dialog box, 1-2-3 creates the following Click event script for the button:

```
Sub Click(Source As ButtonControl)
     .MacroRunText("{macro_command}")
End Sub
```

Note If you open a .WK4 file that contains macros stored with buttons and then subsequently save the file as a .WK4 file, macros stored with buttons are preserved.

Example

The following sub runs a macro that enters the label "Weekly Status Report" in the current cell and changes the font and column width:

```
Sub Click()
     .MacroRunText(| _
          {CELL-ENTER "Weekly Status Report"} _
          {STYLE-FONT-ALL "Times New Roman"; 24} _
          {COLUMN-WIDTH 30} _
     |)
End Sub
```

{button ,AL(`H_123_CREATING_A_BUTTON_STEPS;H_123_MACRORUN_METHOD_MEMDEF;H_123_MACRORUNTEXT_METHOD_MEMDEF',0)} See related topics

Creating a script button

You can create a button on a sheet and attach a script to it. When the user clicks the button, the script runs.

1. Choose Create - Button.

2. Position the mouse pointer in the sheet where you want the button to appear.
3. Do one of the following:
   - To create a button in the default size, click the sheet.
   - To size the button, drag across the sheet and release the mouse button when the button is the size you want.

   1-2-3 selects the button and displays the Script Editor, which contains an empty script for the button's Click event.
4. Create a script that you want to run when a user clicks the button.
5. Choose File - Close Script Editor when you finish writing the script.

{button ,AL(`H_123_CREATING_A_BUTTON_DETAILS',1)} See details
{button ,AL(`H_123_CONVERTING_MACRO_BUTTONS_OVER;',0)} See related topics

Details: Creating a script button

Changing button text
To display a label on the button, other than the default label "Button," select the button and then choose Drawing - Drawing Properties (Basics tab). Enter the new label in the "Text" box.

Anchoring buttons
By default, a button is fastened to the top left and bottom right underlying cells only. To change how a button is fastened to the cells behind it, select the button and then choose Drawing - Drawing Properties (Basics tab).

Hiding and locking buttons
You can hide buttons when you do not want users to see them. You can also lock buttons so that users cannot change their scripts or properties. To hide or lock a button, select the button and then choose Drawing - Drawing Properties (Basics tab).

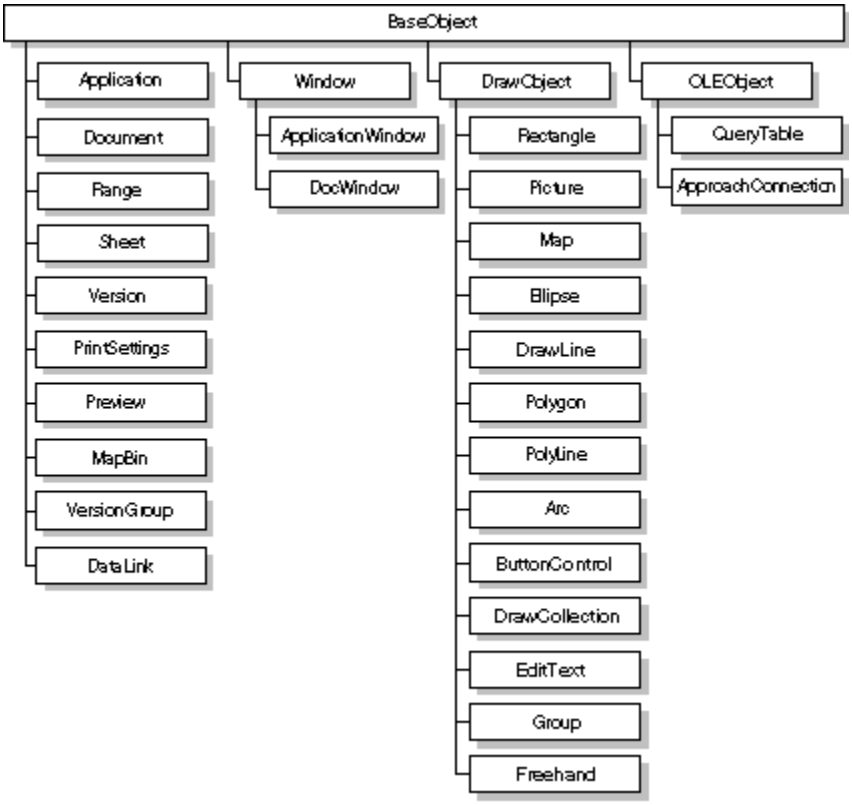
Running a macro from a button
You can run a macro from a button by using either the MacroRun or MacroRunText method in the script for the button.

{button ,AL(`H_123_CREATING_A_BUTTON_STEPS',1)} Go to procedure

{button ,AL(`H_123_CREATING_A_BUTTON_DETAILS_RT;H_STYLING_GRAPHICS_OVER;H_MOVING_GRAPHICS_STEPS;H_NAMING_A_GRAPHIC_STEPS;H_SELECTING_GRAPHICS_STEPS;',0)} See related topics

1-2-3 Inheritance relationships

The following diagram illustrates the most important inheritance relationships in 1-2-3.



Installing the sample script files

The sample script files are not installed by the Install program. If you have the 1-2-3 97 or SmartSuite 97 CD-ROM, you can install the sample script files from there.

Once the sample script files are copied to the correct location on your computer, you can access the sample scripts by opening the file SCRIPTS.123.

Installing the sample script files from the CD-ROM

1. Create a folder named \SCRIPTS in your root folder. For example, C:\SCRIPTS.
2. Insert the CD-ROM in the appropriate drive.
3. Start the Windows Explorer.
4. Double-click the icon for the CD drive.
5. Open the EXTRA folder, the 123 folder, and then the SCRIPTS folder.
6. Select the files SCRIPTS.123 and CURRENCY.123 and choose Edit - Copy.
7. Open the folder you created in the first step and choose Edit - Paste.
8. Double-click the icon for the CD drive.
9. Open the EXTRA folder, the 123 folder, and then the SCRIPTS folder.
10. Select the file SCRIPTS.SMI and all the .BMP files and then choose Edit - Copy.
11. Open the 1-2-3 Icons folder (typically \LOTUS\123\ICONS), and choose Edit - Paste.

Setting up the Sample Scripts set of SmartIcons

1. Start 1-2-3.
2. Choose File - User Setup - SmartIcons Setup.
3. Under Bar to setup, choose Sample Scripts from the "Bar name" list.
4. Select the Always option from the "Bar can be displayed when context is" list.
5. Make sure "Bar is enabled to display during its context" is selected.
6. Click OK.

Note 1-2-3 will not automatically remove these files if you choose to uninstall 1-2-3. Instead, you will need to manually delete these files.

Opening the Dialog Editor
To open the Dialog Editor in 1-2-3, choose Edit - Scripts & Macros - Show Dialog Editor.

{button ,AL(`H_123_OPENING_THE_DIALOG_EDITOR_DETAILS',1)} See details
{button ,AL(`H_LDE_OVERVIEW_THE_LOTUSSCRIPT_DIALOG_EDITOR_OVER',0)} See related topics

Details: Opening the Dialog Editor

Using custom dialog boxes created with previous releases
In previous releases of 1-2-3, after you created a dialog box in the Dialog Editor, you copied it to the Clipboard and pasted it in a worksheet file so that you could use it in a macro. You can continue to run the macros that display these dialog boxes, but you cannot upgrade them.

Related SmartIcons

Displays the Script Editor

{button ,AL(`H_123_OPENING_THE_DIALOG_EDITOR_STEPS',1)} Go to procedure

Overview: Working with the sample scripts

1-2-3 97 includes a collection of sample scripts that illustrate LotusScript application development concepts in 1-2-3. These scripts are stored in the self-documented file SCRIPTS.123. For information about installing SCRIPTS.123 and all other files associated with the sample scripts, see Installing the sample script files.

Opening the Script Editor
To open the Script Editor in 1-2-3, choose Edit - Scripts & Macros - Show Script Editor.

{button ,AL(`H_123_VIEWING_A_SCRIPT_DETAILS',1)} See details
{button ,AL(`H_IDE_THE_LOTUSSCRIPT_IDE_OVER',0)} See related topics

Details: Opening the Script Editor

Other ways to view a script
You can also display the Script Editor by selecting an object, right-clicking it, and then choosing Show Script from the object's shortcut menu. The Script Editor displays the first script associated with the object.

Related SmartIcons

Displays the Dialog Editor

{button ,AL(`H_123_VIEWING_A_SCRIPT_STEPS',1)} Go to procedure

1-2-3: ClosePreview method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Closes the Preview window.

Syntax

application.**ClosePreview**

Parameters

None

Return values

None

1-2-3: GetRangeString method

{button ,AL(`H_123_RANGESELECTOR_CLASS',0)} See list of classes

{button ,AL(`H_123_GETRANGESTRING_METHOD_EXSCRIPT;',1)} See example

Allows the user to use the mouse to point to a range of cells in any open workbook. It returns a string.

Syntax

range **=** *rangeselector*.**GetRangeString**(*document*)

Parameters

document

Document. The document where GetRangeString is invoked this parameter allows the range names to be relative to a particular file.

Return values

String. The range name or address of the range selected when 1-2-3 is in Point mode.

Usage

The string can be used as an input value for a text control or a cell. This parameter allows range names to be relative to a particular file.

```
' Example: GetRangeString method
' Put range string into a variable.
Dim rs As RangeSelector
Dim r as String
Set rs = CurrentApplication.RangeSelector
r = rs.GetRangeString(CurrentDocument)
MessageBox "The selected range is" + r
```

1-2-3: GetRange method

{button ,AL(`H_123_RANGESELECTOR_CLASS',0)} See list of classes

{button ,AL(`H_123_GETRANGE_METHOD_EXSCRIPT;',1)} See example

Allows the user to use the mouse to point to a range of cells in any open workbook. It returns a Range object.

Syntax

range **=** *rangeselector***.GetRange()**

Parameters

None

Return values

Range. The range selected when 1-2-3 is in Point mode.

Usage

When GetRange is called from a LotusScript routine, execution of the script halts, a Range Selector window opens, and 1-2-3 changes to Point mode. The user can then use the mouse to select a range of cells. While the user is dragging, the coordinates of the currently selected cells are displayed in the Range Selector window.

After selecting the range, a range object is returned, and execution of the script resumes.

```
' Example: GetRange method
' An example of changing 1-2-3 to Point mode and
' assiging the returned value to a variable.
Dim MyRange As Range
Set MyRange = CurrentApplication.RangeSelector.GetRange()
```

1-2-3: HelpContents method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Displays the Help topics browser for 1-2-3 help.

Syntax

application.**HelpContents**

Parameters

None

Return values

None

1-2-3: IsAddinLoaded method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Tests if the specified addin has been loaded into memory.

Syntax

boolean **=** *application*.**IsAddinLoaded**(*addinname*)

Parameters

AddinName

String. The name of the addin.

Return values

Variant (Boolean). Returns True if *addinname* is loaded, False if not.

1-2-3: Lock method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Locks the file's formats.

Syntax

document.**Lock** *formatprotection, password*

Parameters

formatprotection

Variant (Boolean). Specifies whether the file's formats are locked (value True) or not (value False).

Password

String. A password associated with the file.

Return values

None

Usage

If you want to lock a file the same way you would using the File - Workbook Properties (Security tab) command, then you must first set the DataProtected property to True and then invoke this method.

1-2-3: LowerRightVisibleCell method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Returns the cell address of the bottom right visible cell in the current sheet.

Syntax

Value **=** *sheet*.**LowerRightVisibleCell**

Parameters

None

Return values

Variant. The cell address of the bottom right visible cell in the current sheet.

1-2-3: MovePoint method

{button ,AL(`H_123_DRAWLINE_CLASS;H_123_FREEHAND_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS',0)} See list of classes

Moves a vertex point in a polyline, polygon, or freehand drawing.

Syntax

object.**MovePoint** *index, horizontal, vertical*

Parameters

index

Long. Specifies a point of a polyline, polygon, or freehand drawing. Any long from 0 (zero) to the total number of points minus one.

horizontal

Long. The number of units in twips to move a point left or right. Specify a positive integer to more the point to the right; specify a negative integer to move the point to the left.

vertical

Long. The number of vertical units in twips to move a point up or down. Specify a positive integer to move the point up; specify a negative integer to move the point down.

Return values

None

1-2-3: TopLeftVisibleCell method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_TOPLEFTVISIBLECELL_METHOD_EXSCRIPT;',1)} See example

Returns the cell address of the top left visible cell in the current sheet.

Syntax

variant **=** *sheet*.**TopLeftVisibleCell**

Parameters

None

Return values

Variant. Thecell address of the top left visible cell in the current sheet.

```
' Example: TopLeftVisibleCell method
' Display the address of the top left visible cell in a message box.
x = .TopLeftVisibleCell.Name
MessageBox x
```

1-2-3: AllFields property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_ALLFIELDS_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the list of all available fields in the source database tables.

Data type

Strings

Syntax

value **=** *query***.AllFields**

Legal values

The values for the AllFields property are the available fields in the source tables.

```
'Example: AllFields property
'Return the list of all available fields in the Output panel.

sub Printfields
Forall x in [Query1].AllFields
Print x," ";
end Forall
End sub
```

1-2-3: CellCommentsFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

{button ,AL(`H_123_CELLCOMMENTSFONT_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the font and the text styles used for printed cell commands.

Data type

Font

Syntax

printsettings.**CellCommentsFont =** *font*

font **=** *printsettings*.**CellCommentsFont**

Legal values

The available fonts are dependent on what you have installed on your system.

```
' Example: CellCommentsFont property
' Apply a font and other attributes to text.
Dim printstyle As PrintSettings
Set printstyle = CurrentDocument.CurrentPrintSettings
printstyle.CellCommentsFont.Fontname = "TimesNewRoman"
printstyle.CellCommentsFont.Bold = True
printstyle.CellCommentsFont.Italic = True
printstyle.CellCommentsFont.DoubleUnderline = True
```

1-2-3: CenterLatitude property

{button ,AL(`H_123_PLOT_CLASS',0)} See list of classes

Sets or returns the center latitude in degrees for the plot area of the map.

Data type

Double

Syntax

mapplot.**CenterLatitude =** *value*

value **=** *MapPlot*.**CenterLatitude**

Legal values

Any double from 0 - 90.

1-2-3: CenterLongitude property

{button ,AL(`H_123_PLOT_CLASS',0)} See list of classes

Sets or returns the center longitude in degrees for the plot area of the map.

Data type

Double

Syntax

mapplot.**CenterLongitude =** *value*

value **=** *mapplot*.**CenterLongitude**

Legal values

Any double from -180 - 180 degrees.

1-2-3: DefaultBackColor property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFAULTBACKCOLOR_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the default background color for new workbooks.

Data type

Color

Syntax

application.**DefaultBackColor =** *color*

color **=** *application*.**DefaultBackColor**

Legal values

For a list of legal values, see the Color palette.

Usage

The DefaultBackColor property is reset when you turn on the UseOSDefaultColors property.

```
Example: DefaultBackColor property
' Set the default cell background color for new workbooks.
CurrentApplication.UseOSDefaultColors = False
CurrentApplication.DefaultBackColor.ColorName = "red"
```

1-2-3: IsSheetHidden property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Sets or returns whether the sheet is hidden.

Data type

Variant (Boolean)

Syntax

sheet.**IsSheetHidden =** *value*

value **=** *sheet*.**IsSheetHidden**

Legal values

Value	Description
TRUE	The sheet is hidden.
FALSE	(Default) The sheet is displayed.

Usage

If there is only one sheet in the file and IsSheetHidden is set to TRUE, then 1-2-3 returns the error: "Cannot hide all sheets."

1-2-3: Language property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_LANGUAGE_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the two character language identification ISO (International Standards Organization) code that specifies the startup language of the application.

Data type

String

Syntax

value **=** *application*.**Language**

Legal values

Any valid two character ISO language code, such as EN for English or FR for French.

```
' Example: Language property
' Display the language ISO code that the application uses in a message box.
LangCode$ = CurrentApplication.Language
MsgBox("The ISO language code is " + LangCode$)
```

1-2-3: Lines property

{button ,AL(`H_123_MAPTITLE_CLASS',0)} See list of classes

(Read-only) Returns the lines of text of the map title.

Data type

MapTextEntries

Syntax

Set *maptextentries* **=** *maptitle*.**Lines**

Legal values

The value for the Lines property is a collection of strings containing the lines of text of the map title.

1-2-3: MenuText property

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

{button ,AL(`H_123_MENUTEXT_PROPERTY_EXSCRIPT;',1)} See example

Returns the text displayed for the menu. This property is read-write for menu commands you create with LotusScript, and read-only for built-in 1-2-3 menus.

Data type

String

Syntax

object.**MenuText =** *text*

text **=** *object*.**MenuText**

Legal values

The value for the MenuText property is a string containing the text displayed for the menu.

Usage

When you enter command names, & (ampersand) followed by a character creates a keyboard shortcut for a command. The letter that follows the & (ampersand) appears underlined; the user can choose this command from the keyboard by pressing ALT plus the underlined letter. For example, if you enter First*Quarter, 1-2-3 displays FirstQuarter; the user can press ALT+Q to select the command. To display & (ampersand) in the command name, enter && (two ampersands). For example, to display B&W, enter B&&W.

```
' Example: MenuText property
' Set the text displayed for a menu.
' The & (ampersand) before the "S" displays as Some Menu (makes S a shortcut key).
object.MenuText = "&Some menu"
```

1-2-3: PaperSizeCustom property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

{button ,AL(`H_123_PAPERSIZECUSTOM_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Determines whether a custom paper size is specified for printing.

Data type

Variant (Boolean)

Syntax

value **=** *printsettings*.**PaperSizeCustom**

Legal values

Value	Description
TRUE	A custom paper size is specified.
FALSE	A custom paper size is not specified.

```
' Example: PaperSizeCustom property
' Put up message box indicating that a custom
' paper size is specified for printing.
If CurrentDocument.CurrentPrintSettings.PaperSizeCustom = TRUE Then
MessageBox "Using custom paper size"
Else
MessageBox "Using standard paper size"
End If
```

1-2-3: PrintSelection property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the selection to be printed, for example, the line named Line1, or the chart named Chart1, or the range A:A2...A:B5.

Data type

Variant (object)

Syntax

Set *printsettings*.**PrintSelection =** *object*

Set *object* **=** *printsettings*.**PrintSelection**

Legal values

A selectable object, such as a range or a graphic object. The object name is enclosed in brackets. For example, [A:B2..A:C3], or [Line 1].

1-2-3: RangeSelector property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_FONT_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns a RangeSelector object.

Data type

RangeSelector

Syntax

rangeselector **=** *application*.**RangeSelector**

Legal values

The value for the RangeSelector property is a RangeSelector object.

Usage

Use the RangeSelector property to return a RangeSelector object and access the object's methods. These methods that let you change 1-2-3 to Point mode and retrieve a range selection.

1-2-3: RowLevel property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

(Read-only) Returns the outline level for the first row in the outlined range.

Data type

Long

Syntax

value **=** *range***.RowLevel**

Legal values

Any long from 0 - 8.

1-2-3: SendOutputToRange property

{button ,AL(`H_123_QUERYTABLE_CLASS',0)} See list of classes

{button ,AL(`H_123_SENDOUTPUTTORANGE_PROPERTY_EXSCRIPT;',1)} See example

Determines or returns whether the result of the query is copied to the range set using the OutputRange property.

Data type

Variant (Boolean)

Syntax

querytable.**SendOutputToRange =** *value*

value **=** *querytable*.**SendOutputToRange**

Legal values

Value	Description
TRUE	Place results of query in range using the OutputRange property.
FALSE	Do not place results of query in range specified using the OutputRange property.

```
' Example: SendOutputToRange property
' Note that this example does not create the query table.

' Copies the results of a query table named Query Table 1
' to the output range A:C18..A:H47.
[Query Table 1].SendOutputToRange = True
[Query Table 1].RestrictOutput = False
Set [Query Table 1].OutputRange = [A:C18..A:H47]
[Query Table 1].RefreshOutput
```

1-2-3: ShowDesignerFrame property

{button ,AL(`;H_123_APPROACHCONNECTION_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

Sets or returns whether to display the designer frame.

Data type

Variant (Boolean)

Syntax

object.**ShowDesignerFrame =** *value*

value **=** *object*.**ShowDesignerFrame**

Legal values

Value	Description
TRUE	(Default) Display the designer frame.
FALSE	Do not display the designer frame.

1-2-3: Target property

{button ,AL(`H_123_DATALINK_CLASS',0)} See list of classes

{button ,AL(`H_123_TARGET_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the location of data from a source file in the destination file.

Data type

Object

Syntax

datalink.**Target =** *value*

value **=** *datalink*.**Target**

Legal values

The value for the Target property is a Range reference.

```
' Example: Target property
' To run this example, substitute a real WordPro filename and bookmark name
' for the filename (LnkTst.lwp) and itemname (test1) in this example.

' Set the datalink, get the range object, set it to G1,
' and go to the range containing the datalink.

Dim datLnk1 As DataLink
Set datLnk1 = CurrentDocument.NewDataLink("Datalink1", _
                              "E:\LnkTst.lwp", "test1", $TextFormat, True)
Set datLnk1.Target = [G1]
Set x = [datLnk1].Target
x.Goto
```

1-2-3: TimeCycle property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_TIMECYCLE_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Specifies whether the operating system is set to display time in a 12- or 24-hour cycle.

Data type

Variant (TimeCycleEnum enumeration)

Syntax

value **=** application**.TimeCycle**

Legal values

Value	Description
$TwelveHour	Display time in 12-hour format.
$TwentyFourHour	Display time in 24-hour format.

```
'  Example: TimeCycle property
Dim ClockType As Variant
ClockType = CurrentApplication.TimeCycle
Msgbox(ClockType)
```

1-2-3: UnitsOfMeasure property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the system of measurement that the operating system uses.

Data type

Variant (UnitsofMeasure enumeration)

Syntax

value **=** *application*.**UnitsOfMeasure**

Legal values

Value	Description
$Metric	Metric units
$Imperial	Imperial units
$Points	Units are measured in points
$Pica	Units are measured in picas
$Twips	Units are measured in twips

1-2-3: BreakLink method

{button ,AL(`H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;H_123_QUERY_CLASS;H_123_APPROACH CONNECTION_CLASS',0)} See list of classes

Breaks the existing link, disconnecting the object from the link's source. Breaking the link converts the OLE object to a Draw object. The object can no longer be edited by the server application.

There is no "undo" for BreakLink; the object is definitively converted to a Draw object.

Syntax

object.**BreakLink**

Parameters

None

Return values

None

Usage

The BreakLink method is equivalent to the Break Link button in the Edit dialog box.

1-2-3: CopyFill method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

Fills a range by copying the data in the range in the specified direction.

Syntax

range.**CopyFill**(*direction*)

Parameters

direction

Variant (CopyFillDir enumeration). Direction in which to copy the data. The following table lists the legal values for this parameter.

Value	Description
$Back	Copy the data to the cells preceding the current location.
$Down	Copy the data to the cells below the current location.
$Forward	Copy the data to the cells after the current location.
$Left	Copy the data to the left.
$Right	Copy the data to the right.
$Up	Copy the data to the cells above the current location.

Return values

None

Usage

The range you specify must include both the cells you want to copy and the cells that you want to fill.

1-2-3: DeleteCurrentVersion method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

Deletes the current version of the specified range.

Syntax

range.**DeleteCurrentVersion**

Parameters

None

Return values

None

1-2-3: GetActiveCell method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_GETACTIVECELL_METHOD_EXSCRIPT ',1)} See example

Returns a range whose top left cell is the active cell.

Syntax

range = *sheet*.**GetActiveCell**

Parameters

None

Return values

Variant. The range whose top left cell is the active cell.

```
' Example: GetActiveCell method
Dim RangeX As Range
Set RangeX = [A].GetActiveCell
```

1-2-3: GetItemText method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

(Read-only) Returns the name of the item at the specified menu position, without the keyboard shortcut. This method returns a blank string as a separator.

Data type

String

Syntax

value = *object*.**GetItemText**(*position*)

Parameters

position

Long. The position in the menu of the item you want to return.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position. If the specified number exceeds the number of menu positions, the GetItemText method returns the last menu item.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position. If the specified number exceeds the number of menu positions, the GetItemText method returns the first menu item.

Return values

String. The name of the menu item at the specified position.

1-2-3: GetItemType method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

(Read-only) Returns the type of the item at the specified position in a menu.

Data type

Variant

Syntax

value **=** *object***.GetItemType**(*position*)

Parameters

position

Long. The position of the item whose type you want to return.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position. If the specified number exceeds the number of menu positions, the GetItemType method returns the last menu item.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position. If the specified number exceeds the number of menu positions, the GetItemType method returns the first menu item.

Return values

Variant (GetItemType enumeration). The type of item at the specified menu position. The following table lists the values for GetItemType.

Value	Description
$Item	Menu item.
$Separator	Menu separator. Separators have no text.
$Menu	Cascade menu.

1-2-3: GotoCirc method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Goes to the first cell containing a circular reference, if there is one.

Syntax

application.**GotoCirc**

Parameters

None

Return values

None

1-2-3: PointX method

{button ,AL(`H_123_DRAWLINE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_FREEHAND_CLASS;',0)} See list of classes

Returns the X coordinate of the specified point of the line, freehand drawing, polyline, or polygon.

Syntax

value **=** *object*.**PointX**(*which*)

Parameters

which

Long. The point whose X coordinate is being returned.

Return values

Long. The X coordinate of the point specified in the *which* parameter.

1-2-3: PointY method

{button ,AL(`H_123_DRAWLINE_CLASS;H_123_FREEHAND_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS',0)} See list of classes

Retuns the Y coordinate of the specified point of the line, freehand drawing, polyline, or polygon.

Syntax

value **=** *object*.**PointY**(*which*)

Parameters

which

Long. The point whose Y coordinate is being returned.

Return values

Long. The Y coordinate of the point specified in the *which* parameter.

1-2-3: RemoveAllVersions method

{button ,AL(`H_123_VERSIONGROUP_CLASS',0)} See list of classes

Removes all the versions from the specified version group.

Syntax

versiongroup.**RemoveAllVersions**

Parameters

None

Return values

None

1-2-3: ResetViewOverrides method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

Resets the view properties for a sheet to the workbook defaults.

Syntax

sheet.**ResetViewOverrides**

Parameters

None

Return values

None

Usage

You can also set the view properties to the workbook defaults using the Sheet - Sheet Properties -View tab.

1-2-3: RestoreToOriginalSize method

{button ,AL(`H_123_PICTURE_CLASS',0)} See list of classes

Restores a picture to its original size.

Syntax

picture.**RestoreToOriginalSize**

Parameters

None

Return values

None

1-2-3: SetLinkSource method

{button ,AL(`H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;H_123_QUERY_CLASS;H_123_APPROACH CONNECTION_CLASS',0)} See list of classes

Sets or returns the source specification for the link.

Syntax

object.**SetLinkSource**([*filename*],[*item*],[*validatesource*])

Parameters

filename

String. (Optional) Specifies the name of the file to use as the link source.

item

String. (Optional) Specifies the item within a file to use as the link source. The syntax for specifying the item depends on the server application.

validatesource

Boolean. (Optional) Specifies whether to check the validity of the file or item. The following table describes the settings for this parameter.

Value	Description
TRUE	Default. Checks the validity of the file or item.
FALSE	Does not check the validity of the file or item.

Return values

Long. The number of characters in the link source specification.

Usage

The SetLinkSource method is similar to the LinkSource property, except that it takes arguments. This allows you to obtain individual parameters separately and to pass them separately. The parameters correspond to the File and Item properties.

1-2-3: SmartSort method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

Sorts data in a range in the specified order.

Syntax

range.**SmartSort**(*sortdirection*)

Parameters

sortdirection

Variant (SortDir enumeration). Describes the direction in which to sort data. The following table shows the values for this parameter.

Value	Description
$Ascend	Sorts from A - Z or from 0 (zero) to the highest number.
$Descend	Sorts from Z - A or from the highest number to 0 (zero).

Return values

None

1-2-3: ToggleVersionBorder method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

Toggles between turning the border on and off for versions of the specified range.

Syntax

range.**ToggleVersionBorder**

Parameters

None

Return values

None

1-2-3: ColumnLevel property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

(Read-only) Returns the outline level for the first column in the range.

Data type

Long

Syntax

value = *range*.**ColumnLevel**

Legal values

The value of the ColumnLevel property is any Long from 0 (zero) to 8.

1-2-3: CurrentPrinter property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the name of the currently selected printer.

Data type

String

Syntax

value **=** *application*.**CurrentPrinter**

Legal values

The value for the CurrentPrinter property is a string containing a printer name.

1-2-3: DateTo21stCentury property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns whether to use the 20th or 21st century in a date that has two digits between 00 - 49 for the year.

Data type

Variant (Boolean)

Syntax

application.**DateTo21stCentury =** *value*

value **=** *application*.**DateTo21stCentury**

Legal values

Value	Description
TRUE	The date is set to the 21st century for year values 00 - 49, so 20 is used for the first two year numbers.
FALSE	The date is set to the 20th century for year values 00 - 49, so 19 is used for the first two year numbers.

1-2-3: DefaultPrintSettings property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the application's default print settings.

Data type

PrintSettings

Syntax

application.**DefaultPrintSettings** = *printsettings*

printsettings = *application*.**DefaultPrintSettings**

Legal values

The DefaultPrintSettings property is determined by the print options found in its PrintSettings object.

1-2-3: Duplex property

{button ,AL(`H_123_PRINTSETTINGS_CLASS;',0)} See list of classes

Determines how the print selection should be printed.

Data type

Variant (DuplexOutput enumeration)

Syntax

printsettings.**Duplex** = *value*

value = *printsettings*.**Duplex**

Legal values

Value	Description
$OneSided	Print on one side of the paper.
$SideToSide	Print on two sides of the paper, with the tops of the pages next to each other.
$EndToEnd	Print on two sides of the paper, with the tops of one page next to the bottom of the following page.

Usage

This property can only be used with printers that support two-sided printing.

1-2-3: FileName property

{button ,AL(`H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;H_123_QUERY_CLASS;H_123_APPROACH CONNECTION_CLASS;',0)} See list of classes

(Read-only) Returns the filename portion of the current link source description.

Data type

String

Syntax

value = *object*.**FileName**

Legal values

The value for the FileName property is the filename part of the current link source.

1-2-3: FindTarget property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns what 1-2-3 searches when finding and replacing labels and values.

Data type

Variant (SpellFindTarget enumeration)

Syntax

application.**FindTarget** = *value*

value = *application*.**FindTarget**

Legal values

Value	Description
$AllWorkbooks	All workbooks
$CurrentWorkbook	The current workbook
$CurrentSheet	The current sheet
$SelectedRange	The selected range

1-2-3: Hidden property

{button ,AL(`H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERY_CLASS;H_123_RECTANGLE_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_QUERYTABLE_CLASS',0)} See list of classes

Sets or returns whether an object is hidden.

Data type

Variant (Boolean)

Syntax

object.**Hidden =** *value*

value **=** *object*.**Hidden**

Legal values

Value	Description
TRUE	The object is hidden.
FALSE	(Default) The object is visible.

1-2-3: IsLinked property

{button ,AL(`H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;H_123_APPROACHCONNECTION_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

(Read only) Returns whether the object is linked or embedded.

Data type

Variant (Boolean)

Syntax

value = *object*.**IsLinked**

Legal values

Value	Description
TRUE	The object is an OLE linked object.
FALSE	The object is an OLE embedded object.

Usage

If the IsLinked property returns the value TRUE, the following methods and properties can be applied to the object: BreakLink, SetLinkSource, Update, AutoUpdate, FileName, and ItemName.

1-2-3: ItemName property

{button ,AL(`H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;H_123_APPROACHCONNECTION_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

(Read-only) Returns the item portion of the current link source description.

Data type

String

Syntax

value = *object*.**String**

Legal values

The value for the ItemName property is the item portion of the current link source. The syntax for the item depends on the server application.

1-2-3: RowHeightUseFontSize property

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_ROWHEIGHTUSEFONTSIZE_PROPERTY_EXSCRIPT',1)} See example

Sets or returns whether to use the font size to determine the row height.

Data type

Variant (Boolean)

Syntax

sheet.**RowHeightUseFontSize =** *value*

value **=** *sheet*.**RowHeightUseFontSize**

Parameters

None

Legal values

Value	Description
TRUE	Use the font size to determine the row height for the sheet.
FALSE	Use the default row height for the sheet.

Usage

The default row height does not adjust to a value that is smaller than the default font size.

```
'Example: DefaultRowHeight and RowHeightUseFontSize properties
Dim testsheet As Sheet
Set testsheet = CurrentDocument.CurrentSheet
testsheet.DefaultRowHeight = 24
testsheet.RowHeightUseFontSize = False
Msgbox "Default row height setting"
testsheet.RowHeightUseFontSize = True
Msgbox "Fit default font setting"
```

1-2-3: ScreenHeight property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the height of the screen display in pixels.

Data type

Long

Syntax

value **=** *application*.**ScreenHeight**

Legal Values

The value for the ScreenHeight property is a long that represents the height of the screen display. The size of the screen display depends on the display settings.

1-2-3: ScreenWidth property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the width of the screen display in pixels.

Data type

Long

Syntax

value **=** *application*.**ScreenWidth**

Legal Values

The value for the ScreenWidth property is a long that represents the width of the screen display. The size of the screen display depends on the display settings.

1-2-3: SheetName property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Sets or returns the name of the sheet in which a specified object is located.

Data type

String

Syntax

sheet.**SheetName =** *value*

value **=** *sheet*.**SheetName**

Legal values

The value for the SheetName property is a string that contains the name of a sheet.

Usage

The SheetName property is like the Name property. However, this property can be found from an object that is not the sheet itself, but is located in the sheet. This provides access to the name of the sheet that provides the location for an object.

1-2-3 ShowDataLossDialog property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns whether a dialog box appears when there is a potential data loss while saving a .WK3 or .WK4 file.

Data type

Variant (Boolean)

Syntax

application.**ShowDataLossDialog =** *value*

value **=** *application*.**ShowDataLossDialog**

Legal values

Value	Description
TRUE	Display a dialog box if there is a potential data loss while saving a .WK3 or .WK4 file.
FALSE	Do not display a dialog box if there is a potential data loss while saving a .WK3 or .WK4 file.

1-2-3: ShowQueryNameAndBorder property

{button ,AL(`H_123_QUERYTABLE_CLASS',0)} See list of classes

Sets and returns whether to display the border and the name of the query table.

Data type

Variant (Boolean)

Syntax

querytable.**ShowQueryNameAndBorder =** *value*

value **=** *querytable*.**ShowQueryNameAndBorder**

Legal values

Value	Description
TRUE	(Default) Display the query table name and border.
FALSE	Do not display the query table name and border.

1-2-3: SortBlanksLast property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets and returns whether 1-2-3 sorts blank cells last.

Data type

Variant (Boolean)

Syntax

application.**SortBlanksLast =** *value*

value **=** *application*.**SortBlanksLast**

Legal values

Value	Description
TRUE	(Default) 1-2-3 sorts blank cells last.
FALSE	1-2-3 does not sort blank cells last.

1-2-3: SortNumbersFirst property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns whether 1-2-3 sorts numbers before words.

Data type

Variant (Boolean)

Syntax

application.**SortNumbersFirst =** *value*

value **=** *application*.**SortNumbersFirst**

Legal values

Value	Description
TRUE	(Default) 1-2-3 sorts numbers before words.
FALSE	1-2-3 does not sort numbers before words.

1-2-3: UpdateSheetDisplay property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns whether 1-2-3 updates the sheet display as changes are made.

Data type

Variant (Boolean)

Syntax

application.**UpdateSheetDisplay =** *value*

value **=** *application*.**UpdateSheetDisplay**

Legal values

Value	Description
TRUE	1-2-3 updates the sheet display as changes are made.
FALSE	1-2-3 does not update the sheet display as changes are made.

1-2-3 VersionName property

{button ,AL(`H_123_VERSION_CLASS',0)} See list of classes

Sets or returns the version part of a Name.

Data type

String

Syntax

version.**VersionName** = *value*

value = *version*.**VersionName**

Legal values

The value for the VersionName property is a string containing the version part of a Version's Name property.

Usage

You refer to a version using a name that is made up of two parts, the range name and the version name.

The Name property is read only and returns the entire name, for example, *range1.version1*. The VersionName property sets or returns only the version part of the name, for example *version.1*.

Developing SmartSuite Applications

Developing SmartSuite Applications Using LotusScript is available in the SmartSuite CD package as an online book. To install *Developing SmartSuite Applications Using LotusScript*, see the SmartSuite installation instructions.

To order a printed version of *Developing SmartSuite Applications Using LotusScript* and other LotusScript user assistance in the SmartSuite 97 Application Developer's Documentation Set, complete the order form and return it to Lotus.

Order Form for the SmartSuite 97 Application Developer's 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/smartsuite/sslotusscript.htm), and available in printed form in the SmartSuite Application Developer's Documentation Set. These books are available only in English.

- *Developing SmartSuite Applications Using LotusScript* is a comprehensive introduction to developing applications for SmartSuite 97. It offers chapters on key programming concepts, using LotusScript programming tools, programming individual SmartSuite products, developing cross-product scripts, and integrating scripts with Notes.
- *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 97 Application Developer's 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, 1997. 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), others must pay by credit card. Please allow 4 - 6 weeks for delivery.

For details on mailing addresses and shipping and handling charges, see fulfillment information in *Getting the Most Out of LotusScript in SmartSuite 97* or on the Lotus World Wide Web home page, http://www.lotus.com/smartsuite/sslotusscript.htm.

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 _____

LotusScript Documentation as Online Books

If you have purchased the CD-ROM version of SmartSuite 97, you can use SmartSuite Install to install the following Online Books about LotusScript:

- *Getting the Most Out of LotusScript in SmartSuite 97*
- *Developing SmartSuite Applications Using LotusScript*
- *LotusScript Language Reference*
- *LotusScript Programmer's Guide*

For more information about installing Online Books, see your SmartSuite 97 installation documentation.

LotusScript Documentation on the World Wide Web

You can view updated versions of LotusScript documentation or download updated sample applications or Help files from the SmartSuite LotusScript home page.

Enter the following URL (Universal Resource Locator) in the location field in your browser and press ENTER:

`http://www.lotus.com/smartsuite/sslotusscript.htm`

Overview: Designing SmartSuite Applications

LotusScript provides a variety of tools and services to support you in developing applications for SmartSuite. Being 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 with the basics

LotusScript applications share the following common features:

- You run script applications in a Lotus product.
- You store scripts in a Lotus product document, such as a 1-2-3 workbook or Word Pro document.
- You use the Lotus Integrated Development Environment (IDE) to edit and debug scripts stored in a product document.
- You use a separate 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 open 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

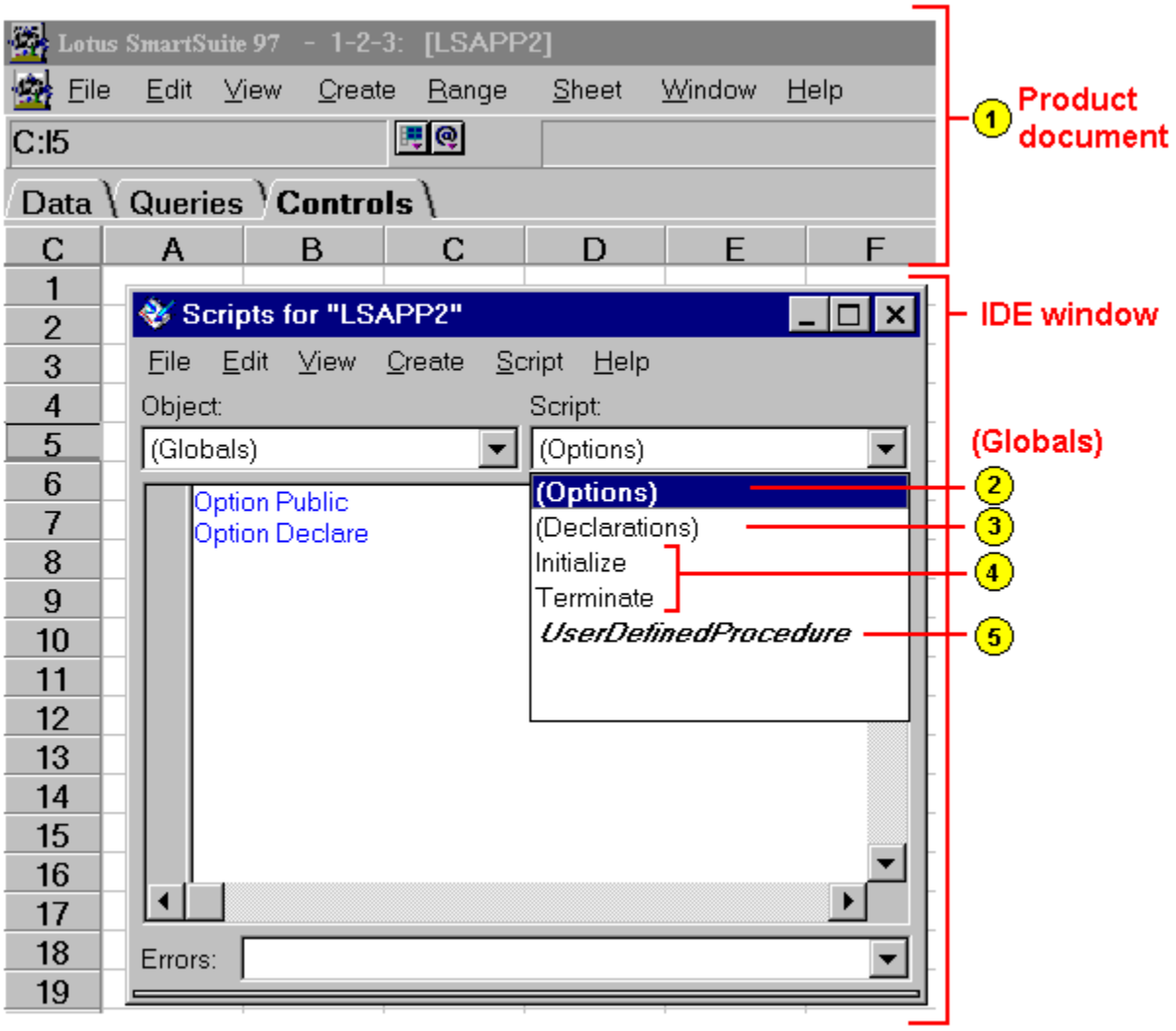
Global scripts make declarations, options, and procedures available to all scripts in your document. For example, to write global scripts for a 1-2-3 document named LSAPP2.123, you must first start 1-2-3, open the document LSAPP2.123, and then open an IDE window for that document. Choose Edit - Scripts & Macros - Show Script Editor in the 1-2-3 menu to open an IDE window for your current document.

The IDE lists objects for which you can write a 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

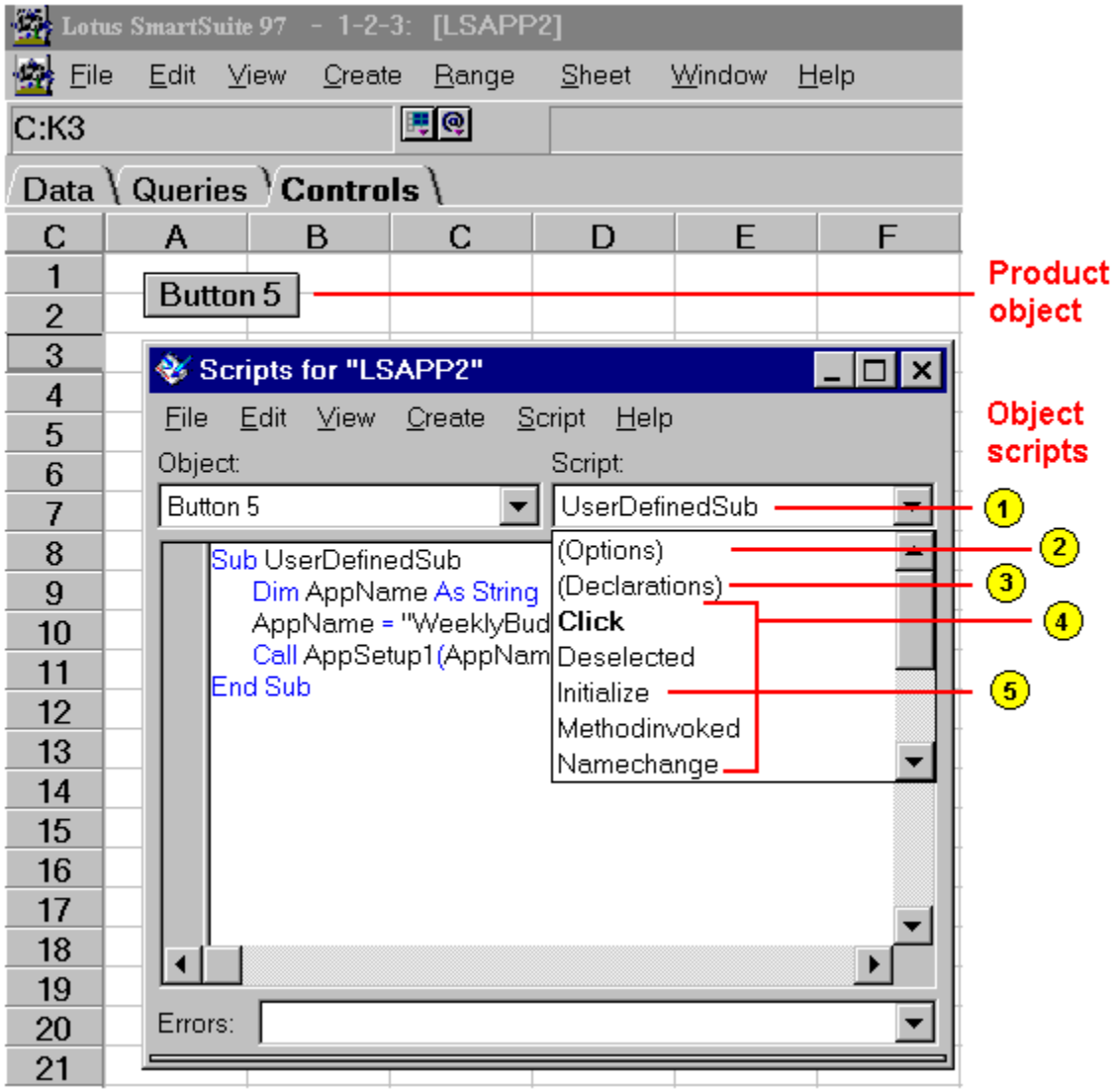
You can also write scripts for product objects in your document. As with (Globals), you can add statements in the predefined scripts for an object or create new procedures for that object. Unlike scripts that you write in (Globals), the declarations, options statements, and procedures that you write 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 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 create this event procedure, select the Button 5 object in the IDE Object list and select 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



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 procedure in the imported script to the Script list.

LotusScript User Assistance for SmartSuite 97

To help you learn how to develop LotusScript applications for SmartSuite 97, Lotus provides a complete library of user assistance. The following books are available in hardcopy, Adobe Acrobat, or HTML formats in your SmartSuite 97 package, in the *SmartSuite Application Developer's Documentation Set*, or on the World Wide Web.

Getting the Most Out of LotusScript in SmartSuite 97

This publication explains how SmartSuite 97 products use the LotusScript programming language and how your business can take advantage of LotusScript in developing applications for SmartSuite.

Developing SmartSuite Applications Using LotusScript

This publication provides comprehensive information on key concepts and techniques for developing LotusScript applications. *Developing SmartSuite Applications Using LotusScript* focuses on programming tools, cross-application programming, Notes integration, and product-specific application development.

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 97 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*.

(Declarations) 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

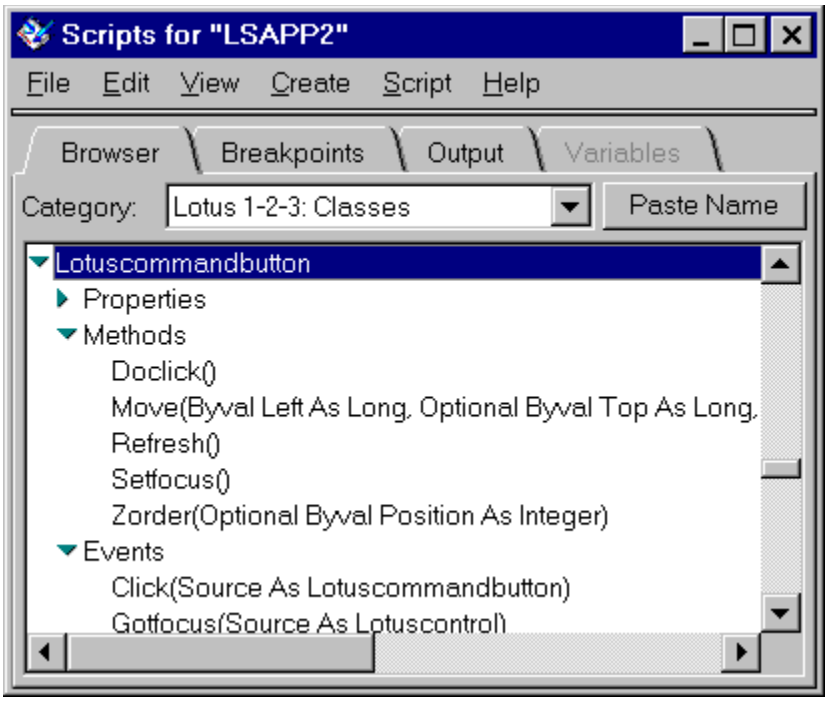
You can create other named subs, functions, and properties for objects in addition to the predefined scripts or event procedures. Because these procedures are not in (Globals), they can be called only from other scripts for the object.

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.

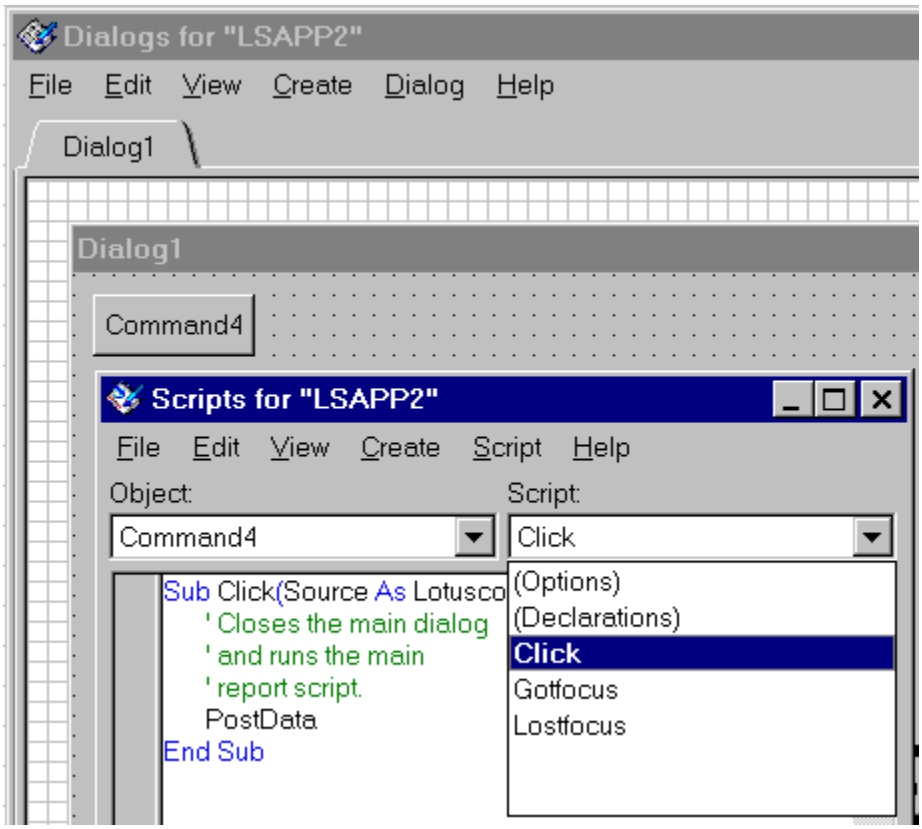
OLE Custom Control (OCX) files

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.



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 a Lotus CommandButton OCX named Command4 are available to you in the IDE.



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
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.

The three types of LotusScript examples are described below. Each type is 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, 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, most examples are not designed 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

FAQs illustrate how to complete common programming tasks using LotusScript. Examples in FAQs not only show how individual statements work but also 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 require you to modify the example to make it run, these dependencies are documented in the Help topic or are noted in comments in the example.

Sample applications

The *Developing SmartSuite Applications Using LotusScript* book includes numerous sample applications for SmartSuite and for individual products. These examples are designed to illustrate more sophisticated tasks for an individual product or tasks that use more than one product. They illustrate how to develop script applications that take advantage of embedded OLE objects, OLE automation, Notes, Visual Basic, the World Wide Web, and custom Dynamic-link libraries (DLLs). Lotus develops new sample applications for SmartSuite on an ongoing basis. These new samples and updated versions of the ones in *Developing SmartSuite Applications Using LotusScript* are available on the World Wide Web.

To copy scripts from these sample applications and paste them into your own script applications, you must first open the sample application document and then display its scripts by opening the Script Editor for that document.

Note All sample applications in *Developing SmartSuite Applications Using LotusScript* are designed to run without modification.

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

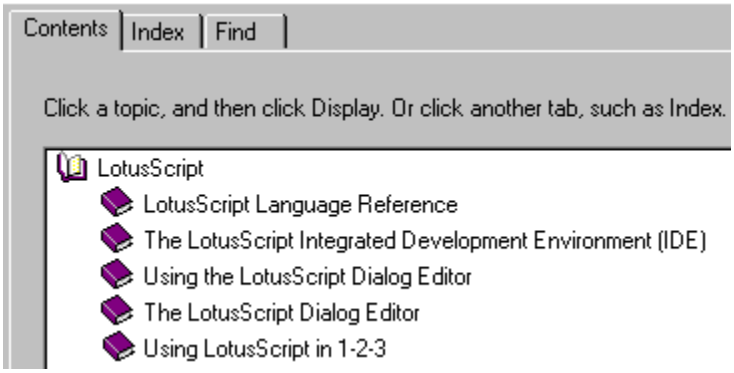
LotusScript Help uses different types of windows to display different types of information, so it is important to know what each type of window contains and how to navigate between them.

Using Help to search for objects and elements

Information in Help is presented in different formats that are designed to assist you in finding general help or information on specific objects and language elements.

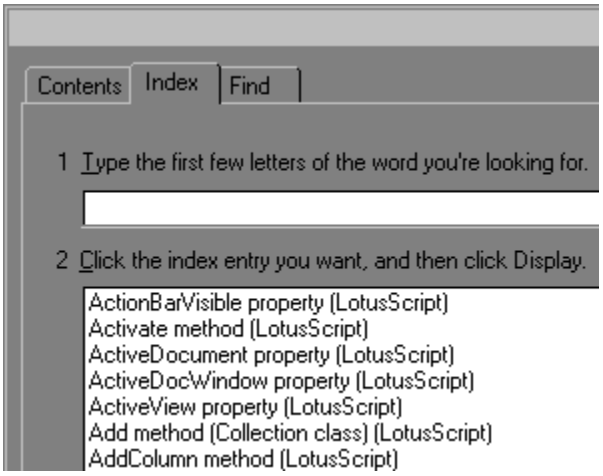
- LotusScript Help Contents

You can use Contents in Help to examine the overall structure of Help and to browse for Help topics relevant to your current script.



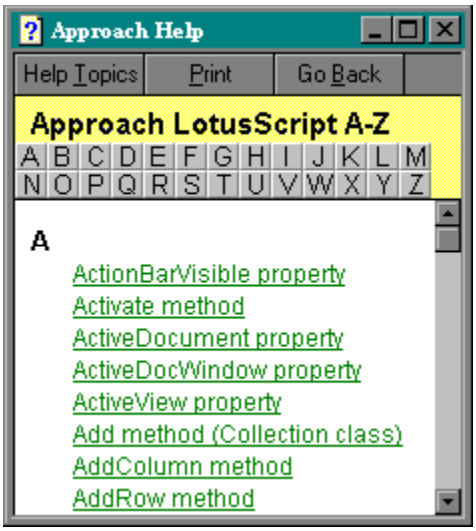
- LotusScript Index

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.



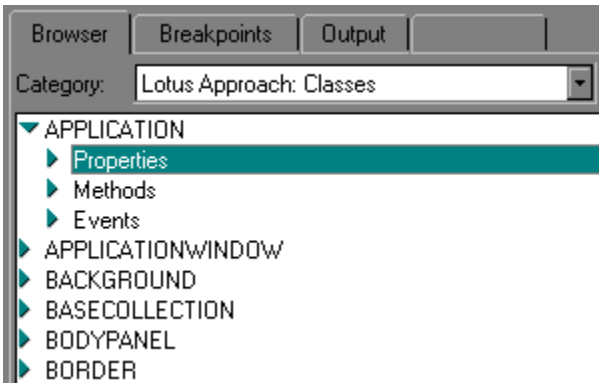
- LotusScript A-Z lists

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.



- 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.



Highlight an element in the Browser panel and press F1 (HELP) to get context-sensitive Help on that element.

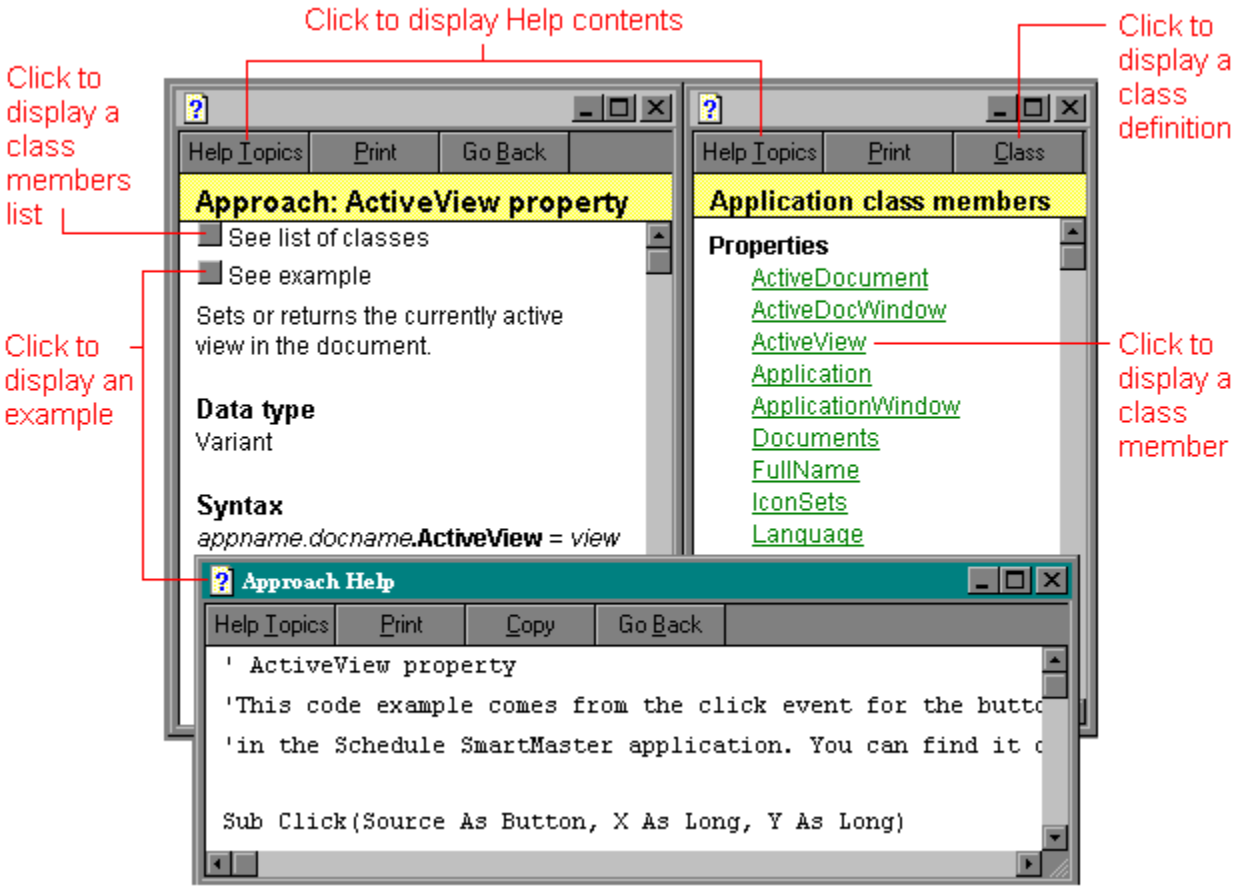
Using Help to write scripts

LotusScript product Help is designed with a focus on classes. As you write scripts, you explore the relationships between product classes and the behaviors of objects in that product.

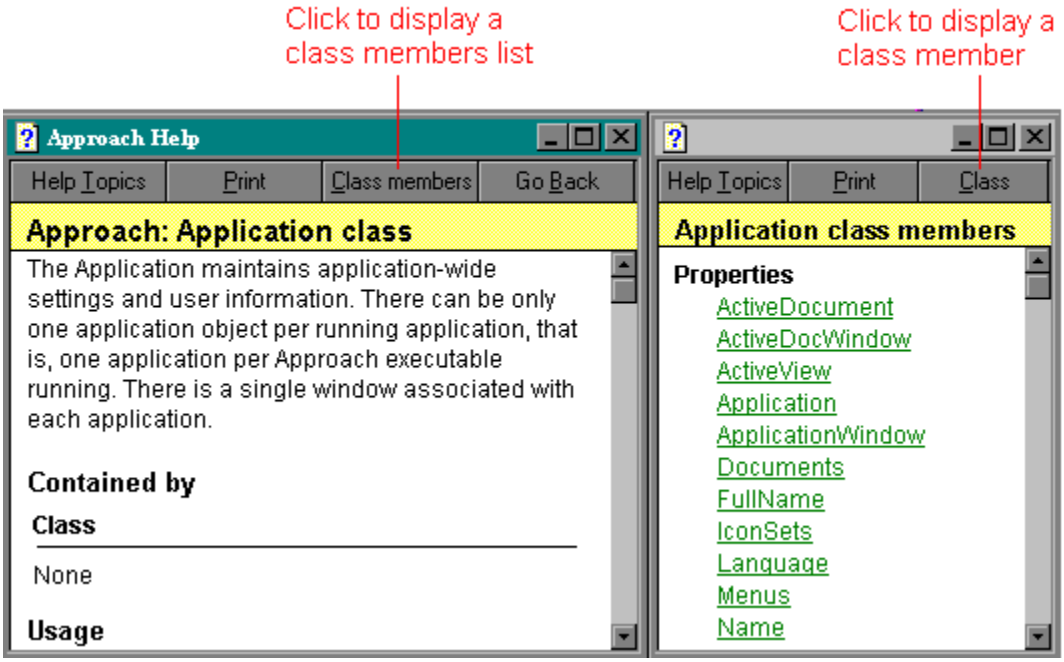
To support this exploration, Help separates information about classes into four topic types:

- Class definition topics 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 topics list all the properties, methods, and events that are members of a particular class.
- Class member topics focus on particular properties, methods, or events.
- Example topics 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 Script Editor.

To navigate among different types of LotusScript Help topics, use the buttons in Help topics and in the Help buttons bar. The following illustration shows how to use buttons to display class member, class member list, and example topics in Help.



The following illustration shows how to display class definition and class member list topics in Help.



Using Help to edit and debug 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, locate the keyboard in the Browser panel, highlight it, 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.

Overview: Fulfillment Information

Refer to the following table when ordering the Application Developer's Documentation Set for SmartSuite 97.

- Identify the fulfillment center/centre for your country
- Copy the mailing address to the front of the fulfillment coupon
- Use the appropriate shipping and handling charge for your country

Currency	*Countries*	*Charge*	*Mailing address*
Australian Dollar	Australia	A$35	Lotus Development Pty Ltd Customer Service Department Level 12, 321 Kent Street Sydney NSW 2000 Australia
Canadian Dollar	Canada	C$15	Lotus Development Corporation SmartSuite Documentation P.O. Box 670 Scarborough, Ontario M1K 5C5
Belgian Franc French Franc Lira Guilder Escudo South African Rand Peseta	Belgium France Italy Netherlands Portugal South Africa Spain	900BF 150F L. 4500 F 50 4500 Esc. R135 3500Pts	Lotus Assistance SARL Parc Club Ariane Bat. Neptune 5 Bld des Chenes, BP 219 78051 St. Quentin en Yvelines Cedex FRANCE
Austrian Schilling Deutchmark Swiss Franc	Austria Germany Switzerland	300 OS 45 DM SFr 35	Lotus Development Gmbh Baierbrunnerstrasse 35 Postfach 70 12 20 81379 Muenchen GERMANY
New Zealand Dollar	New Zealand	NZ$40	Lotus Development New Zealand Ltd Customer Service Dept Level 20, ASB Bank Centre Cnr Albert & Wellesley Sts Auckland New Zealand
US Dollar	United States	$10	Lotus Development Corporation SmartSuite Documentation P.O. Box 25367 Rochester NY 14625-0367 USA
Danish Krone Markka Punt Norwegian Krone Krona Sterling	Denmark Finland Ireland Norway Sweden United Kingdom	Dkr 175 130 mk IR£15 Nkr 190 200 Skr £15	Lotus Development European Corporation Lotus Park The Causeway Staines Middlesex TW18 9AG ENGLAND
US Dollar	Others	US$30.00	Lotus Development Corporation SmartSuite Documentation P.O. Box 25367 Rochester NY 14625-0367 USA

Australia

Lotus Development Pty Ltd
Customer Service Department
Level 12, 321 Kent Street
Sydney NSW 2000
Australia

Canada

Lotus Development Corporation
SmartSuite Documentation
P.O. Box 670
Scarborough, Ontario M1K 5C5

France

Lotus Assistance SARL
Parc Club Ariane

Bat. Neptune 5
Bld des Chenes, BP 219
78051 St. Quentin en Yvelines Cedex
FRANCE

Germany
Lotus Development Gmbh
Baierbrunnerstrasse 35
Postfach 70 12 20
81379 Muenchen
GERMANY

New Zealand
Lotus Development New Zealand Ltd
Customer Service Dept
Level 20, ASB Bank Centre
Cnr Albert & Wellesley Sts
Auckland New Zealand

United Kingdom (U.K.)
Lotus Development European Corporation
Lotus Park
The Causeway
Staines Middlesex TW18 9AG
ENGLAND

United States
Lotus Development Corporation
SmartSuite Documentation
P.O. Box 25367
Rochester NY 14625-0367

Country	*Currency*	*SHCharge*	*Center*
Australia	Australian Dollar	A$30.00	Australia
Austria	Austrian Schilling	30.00 ÖS	Germany
Belgium	Belgian Franc	30.00 BF	France
Canada	Canadian Dollar	C$10.00	Canada
Denmark	Danish Krone	Dkr 30.00	U.K.
Eastern Europe	US Dollar	$30.00	
Finland	Markka	30.00 mk	U.K.
France	French Franc	30.00 F	France
Germany	Deutschmark	30.00 DM	Germany
Ireland	Punt	IR£30.00	U.K.
Italy	Lira	L. 30	France

Luxembourg	Luxembourg Franc	30.00 LF	France
Netherlands	Guilder	F 30.00	France
New Zealand	New Zealand Dollar	NZ$30.00	New Zealand
Norway	Norwegian Krone	Nkr 30.00	U.K.
Portugal	Escudo	30.00 Esc.	France
South Africa	South African Rand	R30.00	France
Spain	Peseta	30 Pts	France
Sweden	Krona	30.00 Skr	U.K.
Switzerland	Swiss Franc	SFr 30.00	Germany
United States	US Dollar	$10.00	United States
U.K.	Sterling	£30.00	U.K.

1-2-3: Application class

Controls the 1-2-3 session.

Base classes

BaseObject

Contained by

Class	Property
All 1-2-3 classes that are not collection classes	Application

Usage

The Application object maintains application-wide settings and user information for a 1-2-3 session. There is only one Application object per running application, that is, one Application object per 1-2-3 executable running. There is a single application window associated with each application. The application window can contain multiple document windows (.123 files).

Use the Application class to perform tasks, such as the following:

- Determine which workbook is active
- Determine the path of the application.
- Set or retrieve the current menu bar

Use the predefined global variable CurrentApplication to refer to the 1-2-3 application in your 1-2-3 scripts. For example, the following statement sets the default path for the 1-2-3 session:

```
CurrentApplication.DefaultPath = "c:\lotus\work\123\"
```

1-2-3: Application class members

Properties

ActiveDocument
ActiveDocWindow
Addins
Application
ApplicationMaximized
ApplicationWindow
ArgumentSeparator
AutoExecMacrosEnabled
AutoOpenPath
AvailableMemory
BeepsOnError
CalcIterations
CalcMode
CalcOrder
CenturyLongFormat
Class
ClassicMenuActivationKey
ClassicMenuEnabled
Colors
ConfirmDragAndDrop
CountryCode
CurrentDirectory
CurrentMenuBar
CurrentPrinter
DateOrder
DateSeparator
DateTo21stCentury
DayNames
DecimalSeparator
DefaultAddinPath
DefaultBackColor
DefaultDecimals
DefaultFileExtension
DefaultFontName
DefaultFontSize
DefaultNegCurrencyFormat
DefaultPath
DefaultPrintSettings
DefaultTextColor
Description
Display4DigitYear
Documents
DragAndDropEnabled
EveningString
FindTarget
FitRowHeightToFont
FullName
GridLineColor
InitialColWidth
InitialRowHeight
Interactive
IsDraggable
IsSelectable
Language
MacroStep

MacroTrace
MatchAccent
MatchKatakana
MatchPitch
MonthNames
Name
NotesPath
NumberOfMostRecentFiles
Parent
Path
PrinterName
PrinterQuality
ProductVersion
RangeSelector
RegisteredCompany
RegisteredUser
ReplaceString
ScreenHeight
ScreenWidth
SearchFormulas
SearchLabels
SearchString
SearchValues
Selection
ShortDayNames
ShortMonthNames
ShowAutomaticPageBreaks
ShowCellCommentMarkers
ShowDataLossDialog
ShowDrawLayer
ShowFormulaMarkers
ShowGridLines
ShowManualPageBreaks
ShowScrollBars
ShowSheetFrame
ShowSheetTabs
ShowVersionBorders
SmartMasterOn
SmartMasterPath
SortBlanksLast
SortDriver
SortNumbersFirst
TabReturnKeyMovement
TextCodepage
TextColumnParseOption
TextColumnParseUserDefined
ThousandsSeparator
TimeCycle
TimeSeparator
TotalMemory
UndoEnabled
UnitsOfMeasure
UpdateLinksOnOpenDoc
UpdateSheetDisplay
UseOSDefaultColors
UserName
UsingTotalToAutoSum
VersionId

Visible
WelcomeOn
Windows
ZoomScale

Methods

Calc
CloseAll
ClosePreview
ColorFromRGB
ExtendedName
FileAdminLinksRefresh
GetEnumString
GetKey
GetMenuPosition
Goto
GotoCirc
HelpContents
IsAddinLoaded
IsSameObject
LoadAddin
NewDocument
NewMenu
NewMenuBar
OpenDocument
OpenDocumentFromNotes
Preview
Print
PrintOut
PrintToFile
Quit
ResetMenuBar
RetrieveFileFromInternet
SendMail
SetInternetOptions
UnloadAddin
UserLogin

Events

Calculate
CancelPrint
DocumentOpened
EndPrint
MenuBarReset
MethodInvoked
NameChange
PropertySet
StartPrint

1-2-3: ApplicationWindow class

The 1-2-3 window. There is one ApplicationWindow object for each 1-2-3 session.

Base classes

Window

Contained by

Class	Property
Application	ApplicationWindow

Usage

The ApplicationWindow class is at the top of the window containment hierarchy for the application.

Use an ApplicationWindow object to perform tasks, such as the following:

- Minimize, maximize, or close the 1-2-3 window
- Hide or display sets of SmartIcons

1-2-3: ApplicationWindow class members

Properties

- Active
- Application
- Caption
- Class
- Description
- EditLineVisible
- Height
- HorizontalScrollBarVisible
- IconBarNames
- IconBarsVisible
- IconSize
- InternetIconsVisible
- IsBubbleHelp
- IsDraggable
- IsSelectable
- Left
- LongPrompt
- Name
- Parent
- StatusBarVisible
- Top
- VersionId
- VerticalScrollBarVisible
- Visible
- Width

Methods

- Activate
- ArrangeIcons
- Cascade
- Close
- ExtendedName
- Goto
- HideIconBar
- IsIconBarShowing
- IsSameObject
- Maximize
- Minimize
- Move
- Resize
- Restore
- ShowIconBar
- Tile
- TileHorizontal
- TileVertical
- Update

Events

- DisplayInit
- GetFocus
- LostFocus
- MethodInvoked
- Moved
- NameChange
- PostClose
- PreClose

PropertySet
Resized

1-2-3: ApproachConnection class

An OLE object that lets you access Approach query table, form, crosstab, report, form letter, and mailing label functionality.

Base classes

OLEObject

Contained by

Class	Property
Document	OLEObjects

1-2-3: ApproachConnection class members

Properties

- Anchor
- Application
- AutoUpdate
- Background
- Class
- Description
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- File
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLinked
- IsLocked
- IsSelectable
- IsSelected
- Item
- Left
- LinkSource
- Name
- Object
- Parent
- ShowDesignerFrame
- Size
- Top
- UserClassNameApplication
- UserClassNameFull
- UserClassNameShort
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- BreakLink
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- SetLinkSource
- ToBack
- ToFront
- Update
- Verb

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Arc class

A graphic object in the shape of an arc.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Use the Arc class to perform tasks, such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Determine the object's background color and pattern.
- Add the object to or remove the object from the current selection

1-2-3: Arc class members

Properties

- Anchor
- Application
- Arrow
- Background
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- Rotation
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- Selected
- PropertySet

1-2-3: Background class

Properties, such as color and pattern, of the background of an object.

Base classes

BaseObject

Contained by

Class	Property
ApproachConnection	Background
Arc	Background
Chart	Background
DrawCollection	Background
EditText	Background
Ellipse	Background
Freehand	Background
Group	Background
Legend	Background
Map	Background
MapPlot	Background
MapTitle	Background
OleObject	Background
Picture	Background
Polygon	Background
Polyline	Background
QueryTable	Background
Range	Background
Rectangle	Background
Sheet	Background

Usage

Use this class to modify the background of any object that has a Background property. A background object cannot be displayed by itself; it is used as the Background property of another object.

1-2-3: Background class members

Properties

- Application
- BackColor
- Class
- Color
- IsDraggable
- IsSelectable
- Name
- Parent
- Pattern
- VersionId

Methods

- ExtendedName
- Goto
- IsSameObject
- RevertToStyle

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: BaseCollection class

The base class for all collection classes in 1-2-3.

Base classes

None

Contained by

None

Usage

The BaseCollection class is an abstract class. That is, you cannot create an instance of BaseCollection. You can, however, represent all its derived classes with the BaseCollection class. For example, you can write a subroutine that takes BaseCollection as a parameter. Then you can use any instance of a derived BaseCollection class in that subroutine:

```
Dim x As BaseCollection
Set x = MyDocument.Ranges
```

1-2-3: BaseCollection class members

Properties

Count

Methods

Item
Next
Open

1-2-3: BaseObject class

The root base class for all classes in 1-2-3 that are not collection classes.BaseCollection is the root base class for all collection classes in 1-2-3.

Base classes

None

Contained by

None.

Usage

The BaseObject class is an abstract class. That is, you cannot create an instance of BaseObject. You can, however, represent all its derived classes with the BaseObject class.

1-2-3: BaseObject class members

Properties

- Application
- Class
- IsDraggable
- IsSelectable
- Name
- Parent
- VersionId

Methods

- ExtendedName
- Goto
- IsSameObject

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: ButtonControl class

A button that, when clicked, calls a LotusScript function or procedure.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Buttons provide quick starts for LotusScript functions or procedures in 1-2-3.

1-2-3: ButtonControl class members

Properties

- Anchor
- Application
- Class
- Font
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- Text
- TextHorizontalAlign
- TextOrientation
- TextRotation
- TextVerticalAlign
- TextWrapped
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Click
- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected
- ValueChange

1-2-3: Chart class

A chart in 1-2-3.

Base classes

BaseObject

Contained by

Class	Property
Document	Charts

Usage

Information on individual Chart classes, properties, and methods is available in LotusChart LotusScript Help. To view these Help topics, choose Help - Help Topics and Double-click the LotusScript book to see the Overview: Using LotusScript in 1-2-3 topic. Double-click the topic to display it. Then choose Chart Classes, Chart Methods, or Chart Properties.

The Charts collection class contains all charts in a workbook.

1-2-3: Chart class members

Properties

- Anchor
- Application
- Background
- Class
- Description
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- ShowDesignerFrame
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Charts class

A collection of Chart objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	Charts

Usage

Iteration is the process of stepping through a collection and acting on each element in the collection. Use the LotusScript ForAll statement to iterate through a collection.

For example, the following statement makes all the charts in the current workbook pie charts:

```
ForAll x in CurrentDocument.Charts
   x.Type = $Pie
End ForAll
```

Indexing is the process of using the Item method or the indexing syntax to access a specific object in the collection. The Item method is a member of every collection class in 1-2-3.

For example, the following statement accesses the second chart in a collection:

```
CurrentDocument.Charts(1)
```

1-2-3: Charts class members

Properties

Count

Methods

Item
Next
Open

1-2-3: ClassInfo class members

Properties

- Application
- Class
- ClassName
- ClassVersionId
- Events
- IsDraggable
- IsSelectable
- Methods
- Name
- Parent
- Properties
- VersionId

Methods

- ExtendedName
- Goto
- IsSameObject

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: ClassInfo class

Provides information about the properties, methods, and events available for instances of a class.

Base classes

BaseObject

Contained by

Class	Property
All 1-2-3 LotusScript classes that are not collection classes	Class

Usage

None of the properties described by ClassInfo can be changed by the user.

1-2-3: ClassInfos class members

This class is reserved by 1-2-3 for internal use.

1-2-3: ClassInfos class

This class is reserved by 1-2-3 for internal use.

1-2-3: Color class

A color assigned to objects that have a color property.

Base classes

BaseObject

Contained by

Class	Property
Application	DefaultBackColor, DefaultTextColor, GridLineColor
ApproachConnection	EdgeColor, FrameColor
Arc	EdgeColor
Background	BackColor, Color
Chart	EdgeColor, FrameColor
Document	GridLineColor
DrawCollection	EdgeColor, FrameColor
DrawLine	EdgeColor
EditText	EdgeColor, FrameColor
Ellipse	EdgeColor
Font	FontColor
Freehand	EdgeColor
Group	EdgeColor; FrameColor
Legend	EdgeColor
Map	EdgeColor; FrameColor
MapBin	Color
MapPlot	EdgeColor
MapTitle	EdgeColor
OLEObject	EdgeColor, FrameColor
Picture	EdgeColor, FrameColor
Polygon	EdgeColor
Polyline	EdgeColor
QueryTable	EdgeColor, FrameColor
Range	FrameColor
RangeBorder	Color
Rectangle	EdgeColor, FrameColor
Sheet	GridLineColor, TabColor

Usage

See the Color palette for a list of the color names in 1-2-3 and their index values.

Use a color object to determine the current levels of red, green, and blue (RGB) that contribute to a specific color. This is useful if you need a precise color for a graphic object such as a corporate logo.

You cannot change the individual RGB components of an existing color, you have to create a new color object and base its RGB settings on the original color.

A color object cannot be displayed; it can only be the Color property of another object. For example, the following code sets the color for the bottom border of the currently selected range:

```
Dim y As Color
Set y = CurrentApplication.Colors("red")
Set [].BottomBorder.Color = y
```

1-2-3: Color class members

Properties

- Application
- Blue
- Class
- ColorIndex
- ColorName
- Green
- IsDraggable
- IsSelectable
- Name
- Parent
- Red
- RGB
- VersionId

Methods

- ExtendedName
- GetRGB
- Goto
- IsSameObject
- SameColor

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: Colors class

A collection of Color objects.

Base classes

BaseCollection

Contained by

Class	Property
Application	Colors

Usage

The Colors collection holds a collection of 240 color constants, representing all the colors in the palette.

1-2-3: Colors class members

Properties

Count

Methods

Item
Next
Open

1-2-3: DataLink class

An OLE link. A link is a channel through which data stored in a source file is displayed in a destination file. When you update linked data in a destination file, the latest data from the source file is displayed. A link is not the same thing as an embedded object.

Base classes

BaseObject

Contained by

Class	Property
Document	DataLinks

Usage

The DataLinks collection class contains all the DataLink objects in a workbook.

1-2-3: DataLink class members

Properties

- Application
- AutoUpdate
- Class
- Description
- FileName
- Format
- IsDraggable
- IsLinked
- IsSelectable
- ItemName
- LinkSource
- Name
- Object
- Parent
- Size
- Target
- UserClassNameApplication
- UserClassNameFull
- UserClassNameShort
- VersionId

Methods

- BreakLink
- ExtendedName
- Goto
- IsSameObject
- SetLinkSource
- Update
- Verb

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: DataLinks class

A collection of DataLink objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	DataLinks

1-2-3: DataLinks class members

Properties

Count

Methods

Item
Next
Open

1-2-3: DateTime class

Controls dates and times in 1-2-3.

Base classes

BaseObject

Contained by

Class	Property
Document	CreationDate, LastPrinted, ModifiedDate
Version	CreationDate, ModifiedDate
VersionGroup	CreationDate, ModifiedDate

Usage

The 1-2-3 DateTime class provides date arithmetic, including converting strings that look like dates and times to LotusScript date objects.

1-2-3: DateTime members

Properties

- Application
- Class
- IsDraggable
- IsLeapYear
- IsSelectable
- L123Seconds
- LocalTime
- LSLocalTime
- Name
- Parent
- VersionId

Methods

- ExtendedName
- Goto
- IsSameObject
- TimeDifference

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: Document class

A 1-2-3 workbook file (.123 file).

Base classes

BaseObject

Contained by

Class	Property
Application	ActiveDocument, Documents
DocWindow	Document

Usage

Use the document class to find information, such as the following:

- File name of the 1-2-3 file
- The name of the author
- A description of the 1-2-3 file
- Creation and modification dates for the 1-2-3 file

Use the predefined global variable CurrentDocument to refer to the workbook that contains the cell pointer. For example, the following code deletes all charts from the current workbook:

```
ForAll x in CurrentDocument.Charts
   x.Clear
End ForAll
```

1-2-3: Document members

Properties

Active
AlwaysReserve
Application
Author
Authors
CalcIterations
CalcMode
CalcOrder
Changed
Charts
Class
CreationDate
CurrentPrintSettings
CurrentSheet
DataLinks
DataProtected
DataQueryNames
Description
DocWindows
DrawnObjects
EditingTime
Embedded
FormatProtected
GridLineColor
HasPassword
IsDraggable
IsSelectable
Keywords
LastEditor
LastPrinted
LastVersionGroup
Location
Maps
ModifiedDate
Name
NamedPrintSettings
NamedRanges
OLEObjects
Parent
Password
Path
QueryTables
RangeHeaderInSort
Ranges
RangeSortHeaderDepth
ReadOnly
Revisions
Revs
Selection
SheetCount
Sheets
ShowAutomaticPageBreaks
ShowCellCommentMarkers
ShowDrawLayer
ShowFormulaMarkers

ShowGridLines
ShowManualPageBreaks
ShowScrollBars
ShowSheetFrame
ShowSheetTabs
ShowVersionBorders
Size
SortRange
Subject
SynchScrolling
Title
VersionId
ViewSplitHeight
ViewSplitStyle
ViewSplitWidth
Zoom
ZoomScale

Methods

Activate
Backsolve
Clear
ClearRangeNames
ClearSplits
Close
Connect
CopySelection
CopyToClipboard
CreateRangeName
CreateRangeNameFromLabel
CreateRangeNameTable
CreateTable
Cut
CutSelection
DeleteNamedPrintSettings
DeleteQuery
DeleteRangeName
Disconnect
Distribution
EndPoll
ExtendedName
ExtendSheetSelectionBack
ExtendSheetSelectionForward
Find
Goto
GroupSheets
IsSameObject
Lock
MatrixInvert
MatrixMultiply
MergeVersions
NewDataLink
NewDocWindow
NewNamedPrintSettings
NewQuery
NewSheet
NewVersionGroup
NextSplit
PageBack

PageForward
Paste
RangeSortDefineKey
RedefineNamedPrintSettings
Regression
RegressionReset
RenameNamedPrintSettings
Replace
ReplaceAll
ReservationGet
ReservationReleased
RetrievePrintSettings
Rollback
Save
SaveAs
SaveAsToInternet
SaveAsToNotes
SaveCopyAs
Send
SendCommand
SendSQL
Show
ShowAllSheets
SortResetKeys
StartPoll
UnGroupSheets
UpdateDefaultPrintSettings
UseDefaultPrintSettings
VersionGroup
VersionGroups
WhatIfTable1
WhatIfTable2
WhatIfTable3
WhatIfTableReset
ZoomIn
ZoomOut
ZoomReset
ZoomTo

Events

CloseWindow
MethodInvoked
NameChange
Opened
OpenWindow
Poll1
Poll2
Poll3
Poll4
PostClose
PostSave
PostSaveAs
PreClose
PreSave
PreSaveAs
PropertySet
SheetChange

1-2-3: Documents class

A collection of Document objects.

Base classes

BaseCollection

Contained by

Class	Property
Application	Documents

Usage

Iteration is the process of stepping through a collection and acting on each element in the collection. Use the LotusScript ForAll statement to iterate through a collection.

For example, the following statement sets the custom view scale for all active workbooks to 75%:

```
ForAll x in CurrentApplication.Documents
   x.ZoomScale = 75
End ForAll
```

Indexing is the process of using the Item method or the indexing syntax to access a specific object in the collection. The Item method is a member of every collection class in 1-2-3.

For example, the following statement accesses the second document in a collection:

```
Application.Documents(1)
```

1-2-3: Documents class members

Properties

Count

Methods

Item
Next
Open

1-2-3: DocWindow class

Manages the window in which 1-2-3 displays an individual file.

Base classes

Window

Contained by

Class	Property
Application	ActiveDocWindow, Windows
Document	DocWindows

Usage

Use the DocWindow class to perform tasks, such as the following:

- Determine which file is displayed in the current window
- Minimize, maximize, or close the 1-2-3 window
- Display or hide horizontal and vertical scroll bars
- Determine whether or not the window is the current window

Use the predefined global variable CurrentWindow to refer to the window in which 1-2-3 displays the current workbook.

1-2-3: DocWindow class members

Properties

- Active
- Application
- Caption
- Class
- Description
- Document
- Height
- HorizontalScrollBarVisible
- IsDraggable
- IsSelectable
- Left
- Name
- Parent
- Top
- VersionId
- VerticalScrollBarVisible
- Visible
- Width

Methods

- Activate
- Close
- ExtendedName
- Goto
- IsSameObject
- Maximize
- Minimize
- Move
- Resize
- Restore
- Update

Events

- GetFocus
- LostFocus
- MethodInvoked
- Moved
- NameChange
- PostClose
- PreClose
- PropertySet
- Resized

1-2-3: DocWindows class members

Properties

Count

Methods

Item
Next
Open

1-2-3: DocWindows class

A collection of DocWindow objects.

Base classes

BaseCollection

Contained by

Class	Property
Application	Windows
Document	DocWindows

1-2-3: DrawCollection class

A user-defined selection of several different types of DrawObject objects.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

A DrawCollection is not a LotusScript collection object. It simply represents a selection that includes multiple DrawObjects.

1-2-3: DrawCollection class members

Properties

- Anchor
- Application
- Arrow
- AutoRedraw
- Background
- BaseMapName
- Class
- ColorBins
- CoordinateRange
- Count
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- KnownRegionAliases
- KnownRegionCodes
- KnownRegionNames
- Left
- Legend
- Name
- Overlays
- Parent
- PatternBins
- Plot
- RegionRange
- Rotation
- Rounded
- ShowDesignerFrame
- TextHorizontalAlign
- TextOrientation
- TextRotation
- TextVerticalAlign
- TextWrapped
- Title
- Top
- VersionId
- Visible
- Width

Methods

- AddOverlay
- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom

Get
Goto
Group
IsSameObject
Move
Paste
RecenterMap
RedrawMap
Remove
RemoveFromSelection
RemoveOverlay
Resize
Select
ToBack
ToFront
UnGroup
ZoomMapIn
ZoomMapOut
ZoomMapReset
ZoomMapTo

Events

Deselected
MethodInvoked
NameChange
PropertySet
Selected

1-2-3: DrawLine class

A graphic object in the shape of a line, arrow, or polyline.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Use the DrawLine class to perform tasks such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Add arrows to the object.
- Add the object to or remove the object from the current selection

1-2-3: DrawLine class members

Properties

- Anchor
- Application
- Arrow
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- EditPoints
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- PointCount
- Rotation
- Top
- VersionId
- Visible
- Width

Methods

- AddPoint
- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom
- Goto
- IsSameObject
- Move
- MovePoint
- Paste
- PointX
- PointY
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: DrawObject class

The base class for all 1-2-3 drawn objects.

Base classes

BaseObject

Contained by

Class	Property
Document	DrawnObjects

Usage

The DrawObject class is an abstract class. That is, you cannot create an instance of DrawObject. The properties, methods and events of the DrawObject apply to all drawn objects inheriting from DrawObject.

The following 1-2-3 classes inherit from the DrawObject class:

- Arc
- ButtonControl
- DrawCollection
- DrawLine
- EditText
- Ellipse
- Freehand
- Group
- Map
- Picture
- Polygon
- Polyline
- Rectangle

The DrawObjects collection contains all drawn objects in a workbook.

1-2-3: DrawObject class members

Properties

- Anchor
- Application
- Class
- Description
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: DrawObjects class

A collection of DrawObject objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	DrawnObjects

1-2-3: DrawObjects members

Properties

Count

Methods

Item
Next
Open

1-2-3: EditText class

A text block.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Use the EditText class to perform tasks, such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Determine the object's background color and pattern.
- Add the object to or remove the object from the current selection
- Change the text the object contains

1-2-3: EditText class members

Properties

- Anchor
- Application
- Background
- Class
- Description
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- Font
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- ShowDesignerFrame
- Text
- TextHorizontalAlign
- TextOrientation
- TextRotation
- TextVerticalAlign
- TextWrapped
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected
- ValueChange

1-2-3: Ellipse class

A graphic object in the shape of an ellipse or a circle.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Use rectangles, rounded rectangles, squares, ellipses, and circles to emphasize data or to create designs such as logos.

Use the Ellipse class to perform tasks, such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Determine the object's background color and pattern.
- Add the object to or remove the object from the current selection

1-2-3: Ellipse class members

Properties

- Anchor
- Application
- Background
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- Rotation
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Font class

The properties of a font.

Base classes

BaseObject

Contained by

Class	Property
ButtonControl	Font
DrawCollection	Font
EditText	Font
Group	Font
Legend	Font
MapTitle	Font
PrintSettings	CellCommentsFont, FooterCenterFont, FooterLeftFont, FooterRightFont, FormulaFont, HeaderCenterFont, HeaderLeftFont, HeaderRightFont
Range	Font
Sheet	Font

Usage

You can modify font properties such as color, size, and name.

Use the Font class to modify text in objects that have a Font property. A font object cannot be displayed by itself; it is used as the font property of another object.

1-2-3: Font class members

Properties

- Application
- Bold
- Class
- DoubleUnderline
- FontColor
- FontName
- IsDraggable
- IsSelectable
- Italic
- Name
- Normal
- Parent
- Size
- Strikethrough
- Underline
- VersionId
- WideUnderline

Methods

- ExtendedName
- Goto
- IsSameObject
- RevertToStyle

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: Freehand class

A graphic object in the shape of a freehand drawing.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Use the Freehand class to perform tasks, such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Determine the object's background color and pattern.
- Add the object to or remove the object from the current selection.

1-2-3: Freehand class members

Properties

- Anchor
- Application
- Arrow
- Background
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- EditPoints
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- PointCount
- Rotation
- Top
- VersionId
- Visible
- Width

Methods

- AddPoint
- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom
- Goto
- IsSameObject
- Move
- MovePoint
- Paste
- PointX
- PointY
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Group class

A set of graphic objects, grouped with the Drawing - Group command or the Group method.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

You can group graphic objects to manipulate and style them as a group rather than individually.

{button ,AL(`H_GROUPING_GRAPHICS_STEPS',0)} See related topics

1-2-3: Group class members

Properties

- Anchor
- Application
- Arrow
- AutoRedraw
- Background
- BaseMapName
- Class
- ColorBins
- CoordinateRange
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- Font
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- KnownRegionAliases
- KnownRegionCodes
- KnownRegionNames
- Left
- Legend
- Name
- Overlays
- Parent
- PatternBins
- Plot
- RegionRange
- Rotation
- Rounded
- ShowDesignerFrame
- TextHorizontalAlign
- TextOrientation
- TextRotation
- TextVerticalAlign
- TextWrapped
- Title
- Top
- VersionId
- Visible
- Width

Methods

- AddOverlay
- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom

- Goto
- Group
- IsSameObject
- Move
- Paste
- RecenterMap
- RedrawMap
- RemoveFromSelection
- RemoveOverlay
- Resize
- Select
- ToBack
- ToFront
- UnGroup
- ZoomMapIn
- ZoomMapOut
- ZoomMapReset
- ZoomMapTo

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Legend class
A map legend.

Base classes
BaseObject

Contained by

Class	Property
Map	Legend

1-2-3: Legend class members

Properties

- Anchor
- Application
- Class
- ColorVisible
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- Height
- Hidden
- InsidePlot
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- PatternVisible
- Placement
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- ExtendedName
- Goto
- IsSameObject
- Move
- RemoveFromSelection
- Resize
- Select

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Map class

A map in 1-2-3.

Base Class

DrawObject

Contained By

Class	Property
Document	Maps

Usage

A map links sheet data to a geographic map. For example, you can use a map to display sales information for each country in the world by linking sales data in the sheet to a map of the world.

The Maps collection class contains all maps in a workbook.

1-2-3: Map class members

Properties

- Anchor
- Application
- AutoRedraw
- Background
- BaseMapName
- Class
- ColorBins
- CoordinateRange
- Description
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- KnownRegionAliases
- KnownRegionCodes
- KnownRegionNames
- Left
- Legend
- Name
- Overlays
- Parent
- PatternBins
- Plot
- RegionRange
- ShowDesignerFrame
- Title
- Top
- VersionId
- Visible
- Width

Methods

- AddOverlay
- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RecenterMap
- RedrawMap
- RemoveFromSelection
- RemoveOverlay
- Resize
- Select

ToBack
ToFront
ZoomMapIn
ZoomMapOut
ZoomMapReset
ZoomMapTo

Events

Deselected
MethodInvoked
NameChange
PropertySet
Selected

1-2-3: MapBin class

A map data bin object.

Base classes

BaseObject

Contained by

Class	Property
Map	ColorBins, PatternBins

Usage

Controls the colors or patterns of data bins for a map. Create a separate MapBin object for each Map object.

The MapBins collection class contains all map data bin objects for a map.

1-2-3: MapBin class members

Properties

- Application
- Class
- Color
- Description
- IsDraggable
- IsSelected
- IsSelectable
- Name
- Parent
- Pattern
- Text
- Value
- VersionId

Methods

- AddToSelection
- ExtendedName
- Goto
- IsSameObject
- RemoveFromSelection
- Select

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: MapBins class
A collection of MapBin objects

Base classes
BaseCollection

Contained by

Class	Property
Map	ColorBins, PatternBins

1-2-3: MapBins class members

Properties

- BinRange
- BinsUsed
- BinType
- Count
- LabelRange
- LabelSource
- StyleRange
- StyleSource
- ValueRange
- ValueSource

Methods

- Item
- Next
- Open

1-2-3: Maps class

A collection of Map objects.

Base classes

BaseCollection

Contained By

Class	Property
Document	Maps

Usage

Iteration is the process of stepping through a collection and acting on each element in the collection. Use the LotusScript ForAll statement to iterate through a collection.

For example, the following statement adds a designer frame to each map in the current workbook:

```
ForAll x in CurrentDocument.Maps
   x.DesignerFrameStyle = $DesignerFrame8
End ForAll
```

Indexing is the process of using the Item method or the indexing syntax to access a specific object in the collection. The Item method is a member of every collection class in 1-2-3.

For example, the following statement accesses the second map in a collection:

```
CurrentDocument.Maps(1)
```

1-2-3: Maps class members

Properties

Count

Methods

Item
Next
Open

1-2-3: MapTextEntries class

The lines of text in a map title.

Base classes

BaseCollection

Contained by

Class	Property
MapTitle	Lines

Usage

To access a single line of text in a title, use the MapTextEntry class.

1-2-3: MapTextEntries class members

Properties

Count

Methods

Item
Next
Open

1-2-3: MapTextEntry class

A single entry in a map title.

Base classes

BaseObject

Contained by

Class	Property
MapTitle	Lines

Usage

You access an individual entry in a line using an index value, such as Lines(1), which represents the first line in a title.

To access the collection of text entries in a title, use the MapTextEntries class.

1-2-3: MapTextEntry class members

Properties

- Application
- Class
- Description
- IsDraggable
- IsSelectable
- IsSelected
- LinkedToCell
- Name
- Parent
- Text
- VersionId

Methods

- AddToSelection
- ExtendedName
- Goto
- IsSameObject
- RemoveFromSelection
- Select

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: MapTitle class

A map title.

Base classes

BaseObject

Contained by

Class	Property
Map	Title

Usage

To access the entries of a map title, use the MapTextEntries class.

1-2-3: MapTitle class members

Properties

- Anchor
- Application
- Background
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- Font
- Height
- Hidden
- IsDraggable
- IsSelectable
- IsSelected
- Left
- Lines
- Name
- Parent
- Placement
- TextHorizontalAlign
- TextOrientation
- TextRotation
- TextVerticalAlign
- TextWrapped
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- ExtendedName
- Goto
- IsSameObject
- Move
- RemoveFromSelection
- Resize
- Select

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Menu class

A menu that you can display in the 1-2-3 main menu.

Base classes

BaseObject

Usage

If you want a menu to be available when the workbook that contains it is active, attach the menu script to the document Opened event:

1. Choose Edit - Scripts & Macros - Show Script Editor.

   The Script Editor appears.
2. Select the name of the current file from the Object list.
3. Select Opened from the Script list.

   The empty Opened sub appears in the Script Editor.
4. Enter statements in the sub that you want to execute when the file opens.
5. Save the script by saving the .123 file.

When you open the file, 1-2-3 automatically runs the script.

If you want a menu to be available at the start of each 1-2-3 session, attach the menu script to the document Opened event, then store the .123 file that contains the script in the "Automatically opened files" directory. At the start of each session, 1-2-3 opens all the files in the "Automatically opened files" directory in alphabetical order, words before numbers. Any scripts attached to the Opened event of a file run when 1-2-3 opens the file.

1-2-3: Menu class members

Properties

- Application
- Class
- Description
- EmbeddedParticipation
- IsDraggable
- IsSelectable
- MenuPrompt
- MenuText
- Name
- Parent
- VersionId

Methods

- AddItem
- AddMenu
- AddSeparator
- CheckItem
- DisableItem
- EnableItem
- ExtendedName
- GetItemText
- GetItemType
- GetMenu
- Goto
- IsSameObject
- RemoveItem
- ReplaceItem
- ReplaceMenu
- UncheckItem

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: MenuBar class

A group of menus displayed across the top of the 1-2-3 application window.

Base classes

BaseObject

Contained by

Class	Property
Application	CurrentMenuBar

1-2-3: MenuBar class members

Properties

- Application
- Class
- Description
- EmbeddedParticipation
- IsDraggable
- IsSelectable
- MenuPrompt
- MenuText
- Name
- Parent
- VersionId

Methods

- AddItem
- AddMenu
- AddSeparator
- CheckItem
- DisableItem
- EnableItem
- ExtendedName
- GetItemText
- GetItemType
- GetMenu
- Goto
- IsSameObject
- RemoveItem
- ReplaceItem
- ReplaceMenu
- UncheckItem

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: MapPlot class

A map plot. A map plot is the area in a map where the geographic data is plotted.

Base classes

BaseObject

Contained by

Class	Property
Map	Plot

Usage

Use the MapPlot class to perform tasks, such as the following:

- Change the position of the map plot
- Maintain map dimensions

1-2-3:MapPlot class members

Properties

- Anchor
- Application
- Background
- CenterLatitude
- CenterLongitude
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- MaintainDimensions
- Name
- Parent
- PlotPosition
- PlotRotation
- Rotation
- Top
- VersionId
- Visible
- Width
- Zoom

Methods

- AddToSelection
- Bounds
- ExtendedName
- Goto
- IsSameObject
- Move
- RemoveFromSelection
- Resize
- Select

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: OLEObject class

An OLE object.

Base classes

BaseObject

Contained by

Class	Property
Document	OLEObjects

Usage

The OLEObjects collection contains all the OLEObject objects in a workbook.

1-2-3: OLEObject class members

Properties

- Anchor
- Application
- AutoUpdate
- Background
- Class
- Description
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- FileName
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLinked
- IsLocked
- IsSelectable
- IsSelected
- ItemName
- Left
- LinkSource
- Name
- Object
- Parent
- ShowDesignerFrame
- Size
- Top
- UserClassNameApplication
- UserClassNameFull
- UserClassNameShort
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- BreakLink
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- SetLinkSource
- ToBack
- ToFront
- Update
- Verb

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: OLEObjects class

A collection of OLEObject objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	OLEObjects

1-2-3: OLEObjects class members

Properties

Count

Methods

Item
Next
Open

1-2-3: Picture class

An imported image displayed in a sheet.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Although you cannot display a picture on a button you create in 1-2-3, you can add a picture, such as a bitmap, to the sheet and then attach a script to the picture. The script runs when the user selects the picture. Attach the script to the picture's Selected event.

1-2-3: Picture class members

Properties

- Anchor
- Application
- Background
- Class
- Description
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- ShowDesignerFrame
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- RestoreToOriginalSize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Polygon class

A graphic object in the shape of a polygon. Polygons are closed shapes consisting of any number of straight or freehand lines.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Use the Polygon class to perform tasks, such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Determine the object's background color and pattern.
- Add the object to or remove the object from the current selection

1-2-3: Polygon class members

Properties

- Anchor
- Application
- Background
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- EditPoints
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- PointCount
- Rotation
- Top
- VersionId
- Visible
- Width

Methods

- AddPoint
- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom
- Goto
- IsSameObject
- Move
- MovePoint
- Paste
- PointX
- PointY
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Polyline class

A graphic object in the shape of a polyline. Polylines are open shapes consisting of any number of straight or freehand lines.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

A polyline is an open shape with straight or freehand line segments.

Use the Polyline class to perform tasks, such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Determine the object's background color and pattern.
- Add the object to or remove the object from the current selection

1-2-3: Polyline class members

Properties

- Anchor
- Application
- Arrow
- Background
- Class
- Description
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- EditPoints
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- PointCount
- Rotation
- Top
- VersionId
- Visible
- Width

Methods

- AddPoint
- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom
- Goto
- IsSameObject
- Move
- MovePoint
- Paste
- PointX
- PointY
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: Preview class

This class is reserved by 1-2-3 for internal use.

1-2-3: Preview class members
This class is reserved by 1-2-3 for internal use.

1-2-3: PrintSettings class

A print style, which is a set of print and page settings for 1-2-3.

Base classes

BaseObject

Contained by

Class	Property
Application	DefaultPrintSettings
Document	CurrentPrintSettings

Usage

To modify a collection of named print styles, use the PrintSettingsCollection class.

To create a PrintSettings object, use the NewNamedPrintSettings method.

1-2-3: PrintSettings class members

Properties

AllPagesPrint
Application
BottomMargin
CellCommentsFont
CellCommentsPrint
CenterLeftToRight
CenterTopToBottom
ChartsPicturesAndDrawPrint
Class
Collate
ColumnTitleRange
Copies
Description
Duplex
FitDrawnObjectToPage
FitToPage
FooterCenter
FooterCenterFont
FooterLeft
FooterLeftFont
FooterRight
FooterRightFont
FormulaFont
FormulasPrint
GridLinesPrint
HeaderCenter
HeaderCenterFont
HeaderLeft
HeaderLeftFont
HeaderRight
HeaderRightFont
IsDraggable
IsSelectable
LeftMargin
Name
Orientation
PaperBinName
PaperBinNames
PaperHeight
PaperHeightMaximum
PaperHeightMinimum
PaperSizeCustom
PaperSizeName
PaperSizeNames
PaperWidth
PaperWidthMaximum
PaperWidthMinimum
Parent
PrintPagesFrom
PrintPagesStart
PrintPagesTo
PrintRange
PrintRangeSaved
PrintSelection
PrintWhat

RightMargin
RowTitleRange
ScalePercent
SheetDataPrint
SheetFramePrint
Stapled
TopMargin
VersionId

Methods

ExtendedName
Goto
IsSameObject

Events

MethodInvoked
NameChange
PropertySet

1-2-3: PrintSettingsCollection class

A collection of PrintSettings objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	NamedPrintSettings

Usage

A collection of PrintSettings objects is equivalent to named print styles for a workbook.

1-2-3: PrintSettingsCollection class members

Properties

Count

Methods

Item
Next
Open

1-2-3: DataQuery class

The information 1-2-3 uses to generate an output range.

Base classes

BaseObject

Contained by

Class	Property
Document	DataQueryNames

Usage

Use the DataQuery class toperform tasks, such as the following:

- The source for input data
- The fields to include in the query
- The criteria to use
- The location for an output range

1-2-3: DataQuery class members

Properties

- AllFields
- AllowsUpdates
- Application
- AutoRefresh
- BaseSourceTable
- Class
- Criteria
- Description
- ExtractingUniqueRecords
- IsDraggable
- IsSelectable
- IsSelected
- Name
- OutputLocation
- Parent
- RecordsLimited
- RecordsLimitMax
- SelectFields
- SQL
- VersionId

Methods

- AddSelectField
- AddToSelection
- CopySQLToClipboard
- CreateComputedField
- DeleteComputedField
- ExtendedName
- FieldAggregateType
- FieldAlias
- GetFieldAlias
- Goto
- IsSameObject
- Join
- QuerySortDefineKey
- Refresh
- RemoveFromSelection
- RemoveSelectField
- ResetFieldAggregates
- Select
- SetRecordsLimitMax
- SortData
- SortReset
- Update

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: QueryTable class

A Lotus Approach object embedded in a 1-2-3 workbook. A query table contains a copy of the records from a source database table in either 1-2-3 or an external table, and is linked to the source database.

Base classes

OLEObject

Contained by

Class	Property
Document	QueryTables

1-2-3: QueryTable class members

Properties

Anchor
Application
AutoUpdate
Background
Class
Description
DesignerFrameStyle
EdgeColor
EdgeDashStyle
EdgeLineWidth
FileName
FrameColor
Height
Hidden
IsDraggable
IsLinked
IsLocked
IsSelectable
IsSelected
ItemName
Left
LinkSource
Name
Object
OutputRange
Parent
RestrictOutput
SendOutputToRange
ShowDesignerFrame
ShowQueryNameAndBorder
Size
Top
UserClassNameApplication
UserClassNameFull
UserClassNameShort
VersionId
Visible
Width

Methods

AddToSelection
Bounds
BreakLink
Clear
CopyToClipboard
Cut
ExtendedName
Goto
IsSameObject
Move
Paste
RefreshOutput
RefreshQuery
RemoveFromSelection
Resize
Select

SetLinkSource
ToBack
ToFront
Update
Verb

Events

Deselected
MethodInvoked
NameChange
PropertySet
Selected

1-2-3: QueryTables class

A collection of QueryTable objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	QueryTables

1-2-3: QueryTables class members

Properties

Count

Methods

Item
Next
Open

1-2-3: Range class

A range.

Base classes

BaseObject

Contained by

Class	Property
Document	Ranges

Usage

1-2-3 does not contain a Cell object: 1-2-3 specifies a cell as a single-cell range.

The Ranges collection contains all ranges in a workbook.

1-2-3: Range class members

Properties

AlignOverColumns
AllNames
Application
Background
BottomBorder
CellComment
CellDisplay
Cells
CellValue
Class
ColumnLevel
ColumnWidth
Contents
CoordinateString
CurrentVersion
Description
DesignerFrameStyle
EndColumn
EndRow
EndSheet
Font
FormatDecimals
FormatName
FrameColor
GridBorder
HorizontalBorder
HorizontalPageBreak
InnerBorder
IsColumnCollapsed
IsColumnHidden
IsDraggable
IsFormatFreqUsed
IsHidden
IsNotesFX
IsParenthesized
IsProtected
IsRangeNamed
IsRowCollapsed
IsRowHidden
IsSelectable
IsSelected
LeftBorder
Name
NegativesInColor
OutlineBorder
Parent
RightBorder
RowHeight
RowLevel
ShowDesignerFrame
StartColumn
StartRow
StartSheet
StyleName
TextHorizontalAlign

TextOrientation
TextRotation
TextVerticalAlign
TextWrapped
TopBorder
VersionBorderVisible
VersionId
VersionStatus
VerticalBorder
VerticalPageBreak

Methods

AddToSelection
AppendRecords
AutoSmartSum
Cell
Clear
CollapseColumn
CollapseRow
CopyFill
CopyToClipboard
Cut
DataParse
DataParseGuess
DefineNamedStyle
DeleteColumns
DeleteCurrentVersion
DeleteNamedStyle
DeleteRecords
DeleteRows
DemoteColumn
DemoteRow
DragAndFill
ExpandColumn
ExpandRow
ExtendedName
Find
FitTallest
FitWidest
FitWidestNumber
Format
FormatReset
FreeCellData
GetCellData
Goto
HideColumns
HideRows
InsertColumns
InsertRows
IsSameObject
MacroRun
ModifyNamedStyle
NewVersion
Paste
PromoteColumn
PromoteRow
QuickCopy
QuickMove
RangeCombine

RangeCombineText
RangeExtract
RangeFill
RangeValue
RecalcRange
RemoveFromSelection
RenameNamedStyle
Replace
ReplaceAll
ReportVersion
ResetColumnWidth
ResetRowHeight
Reshape
RevertToNamedStyle
Select
Send
SetActiveCell
SetCellData
SetGalleryStyle
SmartSort
SmartSum
Sort
StyleFontReset
ToggleVersionBorder
Transpose
UnhideColumns
UnhideRows
Version
Versions

Events

CellContentsChange
CellValueChange
Deselected
MethodInvoked
NameChange
PropertySet
Selected

1-2-3: RangeBorder class

The border of a range.

Base classes

BaseObject

Contained by

Class	Property
Range	BottomBorder, GridBorder, HorizontalBorder, InnerBorder, LeftBorder, OutlineBorder, RightBorder, TopBorder, VerticalBorder

Usage

Use the RangeBorder class to set the color and style of a range border.

1-2-3: RangeBorder class members

Properties

- Application
- Class
- Color
- IsDraggable
- IsSelectable
- Name
- Parent
- Style
- VersionId

Methods

- ExtendedName
- Goto
- IsSameObject

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: Ranges class

A collection of Range objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	Ranges
Range	Cells

Usage

Iteration is the process of stepping through a collection and acting on each element in the collection. Use the LotusScript ForAll statement to iterate through a collection.

For example, the following statement enters in the current row the names of all ranges in the current workbook:

```
ForAll x In CurrentDocument.Ranges
   .Contents =  x.Name
   .MoveCellPointer $down, 1
End ForAll
```

Indexing is the process of using the Item method or the indexing syntax to access a specific object in the collection. The Item method is a member of every collection class in 1-2-3.

For example, the following statement accesses the second range in a collection:

```
CurrentDocument.Ranges(1)
```

You can access any Range object by using the string equivalent of its address as the index into the Ranges collection. For example, the following code changes the background color of the range A:A1..A:B10 to red:

```
CurrentDocument.Ranges("A:A1..A:B10").Background.BackColor.ColorName = "red"
```

1-2-3: Ranges class members

Properties

Count

Methods

Item
Next
Open

1-2-3: RangeSelector class

Provides for user selection of a range while a script is running.

Base classes

BaseObject

Contained by

Class	Property
Application	RangeSelector

Usage

The RangeSelector class provides script writers with an easy way to let users select a range during script execution. For example, the following script lets the user select a range and then changes the font in the selected range:

```
Sub ChangeFont
   Dim rs as RangeSelector
   Dim r as Range
   Set rs = CurrentApplication.RangeSelector
   Set r = rs.GetRange
   r.Font.FontName = "Courier New"
End Sub
```

1-2-3: RangeSelector class members

Properties

- Application
- Class
- IsDraggable
- IsSelectable
- Name
- Parent
- VersionID

Methods

- ExtendedName
- GetRange
- GetRangeString
- Goto
- IsSameObject

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: Rectangle class

A graphic object in the shape of a rectangle or square.

Base classes

DrawObject

Contained by

Class	Property
Document	DrawnObjects

Usage

Use rectangles, rounded rectangles, squares, ellipses, and circles to emphasize data or to create designs such as logos.

Use the Rectangle class to perform tasks, such as the following:

- Hide or lock the object
- Determine the object's edge color, edge style, and edge width
- Determine the object's background color and pattern.
- Add the object to or remove the object from the current selection.

1-2-3: Rectangle class members

Properties

- Anchor
- Application
- Background
- Class
- Description
- DesignerFrameStyle
- EdgeColor
- EdgeDashStyle
- EdgeLineWidth
- FrameColor
- Height
- Hidden
- IsDraggable
- IsLocked
- IsSelectable
- IsSelected
- Left
- Name
- Parent
- Rotation
- Rounded
- ShowDesignerFrame
- Top
- VersionId
- Visible
- Width

Methods

- AddToSelection
- Bounds
- Clear
- CopyToClipboard
- Cut
- ExtendedName
- FlipLeftRight
- FlipTopBottom
- Goto
- IsSameObject
- Move
- Paste
- RemoveFromSelection
- Resize
- Select
- ToBack
- ToFront

Events

- Deselected
- MethodInvoked
- NameChange
- PropertySet
- Selected

1-2-3: RoCollection class

This class is reserved by 1-2-3 for internal use.

1-2-3: RoCollection class members

This class is reserved by 1-2-3 for internal use.

1-2-3: Sheet class

One sheet in a workbook.

Base classes

BaseObject

Contained by

Class	Property
Document	CurrentSheet, Sheets

Usage

The Sheets collection class contains all the sheets in a workbook.

To create a Sheet object, use the NewSheet method.

1-2-3: Sheet class members

Properties

- ActiveCell
- Application
- Background
- Class
- ColumnFolding
- ColumnOutlineVisible
- DefaultColumnWidth
- DefaultRowHeight
- Description
- DisplayZeroAs
- Font
- FormatDecimals
- FormatName
- GridLineColor
- HorizontalTitle
- IsDraggable
- IsFormatFreqUsed
- IsParenthesized
- IsProtected
- IsSelectable
- IsSelected
- IsZeroDisplayed
- Name
- NegativesInColor
- Parent
- RowFolding
- RowHeightUseFontSize
- RowOutlineVisible
- SheetName
- SheetNumber
- ShowDrawLayer
- ShowGridLines
- ShowSheetFrame
- TabColor
- TextHorizontalAlign
- TextOrientation
- TextRotation
- TextVerticalAlign
- TextWrapped
- VersionId
- VerticalTitle
- WindowsDefaultsDisplayed

Methods

- AddToSelection
- ClearOutline
- DeleteSheet
- ExtendedName
- ExtendSelection
- Find
- Format
- FormatReset
- GetActiveCell
- Goto
- HideSheet
- IsSameObject

LowerRightVisibleCell
MacroRunText
MoveCellPointer
MoveOrigin
NewApproachConnection
NewArc
NewArrow
NewButton
NewChart
NewDrawLine
NewEditText
NewEllipse
NewFreehand
NewMap
NewObject
NewPicture
NewPolygon
NewPolyline
NewQueryTable
NewRectangle
NewRoundedRectangle
OutlineColumnsToLevel
OutlineRowsToLevel
RemoveFromSelection
Replace
ReplaceAll
ResetViewOverrides
ScrollToActiveCell
Select
SelectAll
SelectAllSheets
SetHorizontalTitle
SetOrigin
SetVerticalTitle
ShowSheet
TopLeftVisibleCell
TurnTo

Events

Deselected
MethodInvoked
NameChange
PropertySet
Selected

1-2-3: Sheets class

A collction of Sheet objects.

Base classes

BaseCollection

Contained by

Class	Property
Document	Sheets

1-2-3: Sheets class members

Properties

Count

Methods

Item
Next
Open

1-2-3: Strings class

A collection of related strings.

Base classes

BaseCollection

Contained by

Class	Property
Application	Addins, DayNames, MonthNames, PrinterNames, ShortDayNames, ShortMonthNames,
ApplicationWindow	IconBarNames
ClassInfo	Events, Methods, Properties
DataQuery	AllFields
Document	Authors, DataQueryNames, NamedRanges
DrawCollection	KnownRegionAliases, KnownRegionCodes, KnownRegionNames, Overlays
Map	KnownRegionAliases, KnownRegionCodes, KnownRegionNames, Overlays
PrintSettings	PaperBinNames, PaperSizeNames
Range	AllNames

1-2-3: Strings class members

Properties

Count

Methods

Item
Next
Open

1-2-3: Version class

A version.

Base classes

BaseObject

Contained by

Class	Property
Range	CurrentVersion

Usage

The Versions collection class contains all versions in a workbook.

1-2-3: Version class members

Properties

- Application
- Author
- Class
- CreationDate
- Description
- IsDraggable
- IsNew
- IsSelectable
- LastEditor
- ModifiedDate
- Name
- Parent
- Share
- StylesRetained
- VersionId
- VersionName

Methods

- DeleteVersion
- ExtendedName
- Goto
- IsSameObject
- MakeCurrent

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: VersionGroup class

A version group.

Base classes

BaseObject

Contained by

Class	Property
Document	LastVersionGroup

Usage

Although the VersionGroup class is a collection of versions associated with a single named range, it is not a collection object. Its Versions method, however, returns a collection that is a selected subset of all the versions in the version group.

1-2-3: VersionGroup class members

Properties

- Application
- Author
- Class
- CreationDate
- Description
- IsDraggable
- IsNew
- IsSelectable
- LastEditor
- ModifiedDate
- Name
- Parent
- Share
- VersionId

Methods

- AddVersion
- DeleteVersionGroup
- ExtendedName
- Goto
- IsSameObject
- MakeCurrent
- RemoveAllVersions
- RemoveVersion
- Versions

Events

- MethodInvoked
- NameChange
- PropertySet

1-2-3: VersionGroups class
A collection of VersionGroup objects.

Base classes
BaseCollection

1-2-3: VersionGroups class members

Properties

Count

Methods

Item
Next
Open

1-2-3: Versions class

A collection of Version objects.

Base classes

BaseCollection

Usage

The Versions class is a collection of all versions associated with a named range; it is not a version group.

1-2-3: Versions class members

Properties

Count

Methods

Item
Next
Open

1-2-3: Window class

A window in 1-2-3. The Window class is a base class for the creation of window objects such as the ApplicationWindow and the DocWindow.

Base classes

BaseObject

Usage

The Window class is an abstract class. That is, you cannot create an instance of Window. You can, however, represent all of the subclasses of the Windows class with the Window class. For example, you can write a subroutine that takes Window as a parameter.Then, you can pass any instance of a Window subclass to that subroutine:

```
Dim x as Window
Set x = CurrentApplication.CurrentDocWindow
```

1-2-3: Window class members

Properties

- Active
- Application
- Caption
- Class
- Description
- Height
- HorizontalScrollBarVisible
- IsDraggable
- IsSelectable
- Left
- Name
- Parent
- Top
- VersionId
- VerticalScrollBarVisible
- Visible
- Width

Methods

- Activate
- Close
- ExtendedName
- Goto
- IsSameObject
- Maximize
- Minimize
- Move
- Resize
- Restore
- Update

Events

- GetFocus
- LostFocus
- MethodInvoked
- Moved
- NameChange
- PostClose
- PreClose
- PropertySet
- Resized

1-2-3: Windows class
A collection of Window objects.

Base classes
BaseCollection

1-2-3: Windows class members

Properties

Count

Methods

Item
Next
Open

1-2-3: Calculate event

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_CALCULATE_EVENT_EXSCRIPT',1)} See example

Occurs when a recalculation ends.

Internal syntax

Calculate(*source*)

Parameters

source

Application. The object on which the event occurred.

{button ,AL(`H_123_CELLVALUECHANGE_EVENT_MEMDEF',0)} See related topics

```
' Example: Calculate event handler
' This handler fits a column's width to accommodate recalculated data.
Sub Calculate(source As Application)
   ' Fit the width of column B to the widest numerical data.
   [A:B4].FitWidestNumber
   ' Make sure the new width isn't too small for the column heading.
   If [A:B4].ColumnWidth < 10 Then
      [A:B4].ColumnWidth = 10
   End If
End Sub
```

1-2-3: CancelPrint event

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTPRINT_ENDPRINT_CANCELPRINT_EVENT_EXSCRIPT',1)} See example

Occurs whens the printing of a document is canceled.

Internal Syntax

CancelPrint(*source*)

Parameters

source

Application. The object on which the event occurred.

1-2-3: CellContentsChange event

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_CELLCONTENTSCHANGE_EVENT_EXSCRIPT',1)} See example

Occurs when the contents of any cell or cell comment in the range changes.

Internal syntax

CellContentsChange(*source*)

Parameters

source

Range. The range on which a Contents property changed.

{button ,AL(`H_123_CELLVALUECHANGE_EVENT_MEMDEF;H_123_VALUECHANGE_EVENT_MEMDEF;H_123_CALCULATE_EVENT_MEMDEF;H_123_CONTENTS_PROPERTY_MEMDEF;H_123_PROPERTYSET_EVENT_MEMDEF',0)} See related topics

```
' Example: CellContentsChange event handler
' Check if the new cell contents remain acceptable.
' Display warning if the contents aren't numeric.
Sub CellContentsChange (source As Range)
   ' Loop over the cells in the source range.
   Forall sglcell In source.Cells
      ' Check whether the cell contents are numeric.
      If IsNumeric(sglcell.Contents) = False Then
         MessageBox "Warning: Cell " & sglcell.CoordinateString _
            & " does not contain a number.", MB_ICONQUESTION, "Script 1"
      End If
   End Forall
End Sub
```

1-2-3: CellValueChange event

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_CELLVALUECHANGE_EVENT_EXSCRIPT',1)} See example

Occurs when the value of any cell within the range changes.

Internal syntax

CellValueChange(*source*)

Parameters

source

Range. The range on which a CellValue property changed.

{button ,AL(`H_123_CELLCONTENTSCHANGE_EVENT_MEMDEF;H_123_CELLVALUE_PROPERTY_MEMDEF;H_123_CALCULATE_EVENT_MEMDEF;H_123_PROPERTYSET_EVENT_MEMDEF',0)} See related topics

```
' Example: CellValueChange event handler
' Check if the new cell value remains valid.
' Display a warning if the value is too large.

' Global declarations
Dim maxValue As Long
' You could run the following sub in the Document.Opened event handler, for example.
' This sub sets a maximum value for the range.
Sub SetMaxValue
   maxValue = 100
End Sub

' Bind the following handler to the CellValueChange event.
Sub CellValueChange (source As Range)
   ' Loop over the cells in the source range.
   Forall sglcell In source.Cells
      ' Check whether the cell value is valid.
      If Not IsNumeric(sglcell.CellValue) Then
         MessageBox "Warning: Cell " & sglcell.CoordinateString _
            & " is not numeric.", MB_ICONQUESTION, "Script 1"
      ElseIf sglcell.CellValue > maxValue Then
         MessageBox "Warning: Cell " & sglcell.CoordinateString _
            & " exceeded the maximum value.", MB_ICONQUESTION, "Script 1"
      End If
   End Forall
End Sub
```

1-2-3: Click event

{button ,AL(`H_123_BUTTONCONTROL_CLASS',0)} See list of classes

{button ,AL(`H_123_CLICK_EVENT_EXSCRIPT',1)} See example

Occurs when the ButtonControl object is clicked.

Internal syntax

Click(*source*)

Parameters

source

ButtonControl. The object on which the event occurred.

{button ,AL(`H_123_GETFOCUS_EVENT_MEMDEF;H_123_SELECTED_EVENT_MEMDEF',0)} See related topics

```
' Example: Click event handler
' Handler for a button that sets the column width or other styles of the current cell.
Sub Click (source As ButtonControl)
   ' Set the column width of the active cell.
   CurrentDocument.CurrentSheet.ActiveCell.Cells(0).ColumnWidth = 15
   ' Set other styles here, if needed.
End Sub
```

1-2-3: CloseWindow event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CLOSEWINDOW_EVENT_EXSCRIPT',1)} See example

Occurs when a document window closes.

Internal syntax

CloseWindow(*source*)

Parameters

source

The Document object whose window closed.

{button ,AL(`H_123_OPENWINDOW_EVENT_MEMDEF;H_123_DOCUMENTOPENED_EVENT_MEMDEF;H_123_OPENED_EVENT_MEMDEF;H_123_PRECLOSE_EVENT_MEMDEF;H_123_POSTCLOSE_EVENT_MEMDEF',0)} See related topics

```
' Example: CloseWindow event handler and ApplicationWindow property.
' Tile the remaining open windows when a window closes.

Sub CloseWindow(source As Document)
   CurrentApplication.ApplicationWindow.TileHorizontal
End Sub
```

1-2-3: Deselected event

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_QUERY_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_PLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_DESELECTED_EVENT_EXSCRIPT',1)} See example

Occurs when the object is removed from the selection.

Internal syntax

Deselected(*source*)

Parameters

source

The object that was removed from the selection.

Usage

The selection is not guaranteed while a Deselected event script runs.

{button ,AL(`H_123_SELECTED_EVENT_MEMDEF;H_123_LOSTFOCUS_EVENT_MEMDEF;H_123_GETFOCUS_EVENT_MEMDEF',0)} See related topics

```
' Example: Deselected event handler.
' Set a flag to indicate that a resource is not needed because of the deselection.

' In the globals section, declare the flag.
Dim resource1Keep As Variant

Sub Deselected(source As OLEObject)
   resource1Keep = False
End Sub
```

1-2-3: DisplayInit event

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_DISPLAYINIT_EVENT_EXSCRIPT',1)} See example

Occurs when a menu bar is first accessed.

Internal Syntax

DisplayInit(*source*)

Parameters

source

The ApplicationWindow object in which the event occurred.

Return values

None

Usage

This event only occurs when the menu bar is first accessed. Subsequently accessing the menu bar doesn't generate this event again.

You can use this event to disable menus.

{button ,AL(`H_123_CLICK_EVENT_MEMDEF;H_123_OPENED_EVENT_MEMDEF;H_123_DOCUMENTOPENED_EVENT_MEMDEF',0)} See related topics

```
' Example: DisplayInit and Opened event handlers; GetMenu and DisableItem methods.
' Disable a menu item when the menu bar is first accessed.

' In the Globals section, declare the application window for the event.
   Dim appWindow As ApplicationWindow

Sub DisplayInit(source As ApplicationWindow)
   ' Disable the first item in the third-from-last menu on the menu bar.
   Dim menu1 As Menu
   Set menu1 = CurrentApplication.CurrentMenuBar.GetMenu(-3)
   menu1.DisableItem 1
End Sub

' Bind the handler in the Opened event on the document.
Sub Opened(source As Document)
   Set appWindow = CurrentApplication.ApplicationWindow
   On Event DisplayInit From appWindow Call DisplayInit
End Sub
```

1-2-3: DocumentOpened event

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DOCUMENTOPENED_EVENT_EXSCRIPT',1)} See example

Occurs when a document is opened during a 1-2-3 session.

Internal Syntax

DocumentOpened(*source*)

Parameters

source

Application. The application that opened the document.

Usage

This event is raised whenever you open a document during a 1-2-3 session. This is different from the Opened event in the Document class, which is only raised when the file containing the Opened event handler script is opened.

If you open more than one document and an event handler script is attached to each document's Opened event, the events are raised in the reverse order that the documents were opened.

```
' Example: DocumentOpened event handler
' Write the current date and time to cell A:A1 whenever
' the document opens. If subsequent documents are opened
' before the current document is closed, the current date
' and time is written to cell A:A1 in each additional
' document as well.

Sub DocumentOpened(source As Application)
   [A:A1].Select
   Selection.Contents = Now
End Sub
```

1-2-3: EndPrint event

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTPRINT_ENDPRINT_CANCELPRINT_EVENT_EXSCRIPT',1)} See example

Occurs when 1-2-3 finishes sending a file to a printer.

Internal Syntax

EndPrint(*source*)

Parameters

source

Application. The object on which the event occurred.

Usage

EndPrint occurs when a document has been successfully sent to the printer queue. This doesn't necessarily mean that the print job has finished.

1-2-3: GetFocus event

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_GETFOCUS_EVENT_EXSCRIPT;H_123_LOSTFOCUS_EVENT_EXSCRIPT',1)} See example

Occurs when the window gets the focus.

Internal syntax

GetFocus(*source*)

Parameters

source

The object that received the focus.

Usage

You can use this event on a DocWindow object to bind handlers for events that are specific to the document or document window.

A window that gets the focus also becomes the active window.

{button ,AL(`H_123_LOSTFOCUS_EVENT_MEMDEF;H_123_DESELECTED_EVENT_MEMDEF;H_123_SELECTED_EVENT_MEMDEF',0)} See related topics

```
' Example: DocWindows, Count properties; GetFocus, LostFocus, Opened,
'          OpenWindow, and CloseWindow event handlers
' Display custom menus for this document whenever a document window gets focus.
' This example handles any number of document windows.
' Note: This example has several hundred lines of code.


' Since the GetFocus event is only raised on a window object,
' you need to set it up explicitly for each DocWindow object that is created,
' rather than for the document as a whole.

' In the Options section for this module, make all declarations Public.
Option Public

' In the Globals section for this module, declare the flags, menu,
' and doc window lists.
Dim isMenu1Up As Variant               ' Flag to prevent adding menu1 twice
Dim isMenu1Made As Variant             ' Flag to prevent creating menu1 twice
Dim menu1 As Menu                      ' Custom menu to add to the menu bar
Dim multiDocWindows List As DocWindow  ' Managed document windows
Dim regFlags List As Integer           ' Boolean to track document windows
registration
Dim windowsCount As Long               ' Number of document windows open
Dim settingValue As Variant            ' Boolean property set by the menu item
Dim settingPos As Integer              ' Menu position of the property item

' Global GetFocus event handler to display the custom menus for this document
' whenever it gets focus.
Sub GetFocus(source As DocWindow)
   ' Add the menu item. (The routine checks to avoid duplication.)
   AddAMenu
End Sub

' Global LostFocus event handler to remove the custom menus for this document
' whenever it loses focus.
Sub LostFocus(source As DocWindow)
   ' The routine will check the menu flag.
   RemoveAMenu
End Sub

' Global handler script for the custom menu item.
Sub itemHandler1
   ' Set the property.
   settingValue = Not settingValue
   ' Set the menu item check mark.
   If settingValue = True  Then menu1.CheckItem   settingPos
   If settingValue = False Then menu1.UncheckItem settingPos
End Sub

' Global sub to bind the GetFocus and LostFocus event handlers to the document window.
Sub RegFocus(docWindow1 As DocWindow, focusRegd As Integer)
   ' Don't bind twice -- the handler will get called twice.
   If focusRegd Then Exit Sub
   ' Bind both events on this window.
   On Event GetFocus From docWindow1 Call GetFocus
   On Event LostFocus From docWindow1 Call LostFocus
   ' And set the flag that says so.
   focusRegd = True
```

```
End Sub

Sub UnregFocus(docWindow1 As DocWindow, focusRegd As Integer)
   ' Remove the GetFocus and LostFocus event handlers from the doc window.
   ' Don't remove the handlers if they're not bound.
   If Not focusRegd Then Exit Sub
   ' Do the On Event ... Remove statements.
   On Event GetFocus From docWindow1 Remove GetFocus
   On Event LostFocus From docWindow1 Remove LostFocus
   ' And indicate that this window is unregistered.
   focusRegd = False
End Sub

Sub AddAMenu
   If isMenu1Up = False Then
      ' Add the custom menu to the menu bar.
      CurrentApplication.CurrentMenuBar.AddMenu -3, menu1
      ' Set the flag indicating that this menu is up.
      isMenu1Up = True
   End If
End Sub

Sub RemoveAMenu
   If isMenu1Up = True Then
      CurrentApplication.ResetMenuBar
      isMenu1Up = False
   End If
End Sub

Sub RegAllDW(doc As document)
   ' Register all open document windows into a window list and into a flag list.
   Dim s$
   ' Loop over all document windows on the document.
   ForAll dw In doc.DocWindows
      ' The list tag will be the window name.
      s$ = dw.Name
      ' If the window isn't in the current list, enter it.
      If Not Iselement(multiDocWindows(s$)) Then
         ' Add the window.
         Set multiDocWindows(s$) = dw
         ' And create a flag for registration, and set it to False.
         regFlags(s$) = False
      End If
      ' Now register this window.
      ' Call by reference so that the real flag gets changed.
      RegFocus multiDocWindows(s$), regFlags(s$)
   End ForAll
End Sub

Sub UnregAllDW(doc As document)
   ' Clean up the menu before abandoning the focus checking.
   RemoveAMenu
   Dim s$
   ' Check all windows on this document.
   ForAll dw In doc.DocWindows
      ' The list tag is the window name.
      s$ = dw.Name
      ' Unregister from the list, including the list of registration flags.
      ' Call by reference so that the real flag will be changed.
```

```
      UnRegFocus multiDocWindows(s$), RegFlags(s$)
   End ForAll
End Sub

Sub MakeMenu1
   If isMenu1Made Then Exit Sub
   isMenu1Up = False
   ' Create the custom menu.
   Set menu1 = CurrentApplication.NewMenu
   ' Set up display text for the custom menu.
   menu1.MenuText = "&Autotasks"
   menu1.MenuPrompt = "Custom automated tasks"
   ' Add items to the menu here ...
   menu1.AddItem 1, "Menu pick 1", "Prompt 1", ThisDocument, "itemHandler1"
   ' Initialize the item settings.
   settingValue = False
   settingPos = 1
   isMenu1Made = True
End Sub

' Handlers for document events.

' In the Opened event on the document, build and display the custom menu
' and bind the focus event handlers to the first DocWindow.
Sub Opened(source As Document)
   ' Make sure there is a menu 1.
   MakeMenu1
   ' Add menu 1 to the menu bar in the third-from-last position.
   AddAMenu
   ' Bind the focus event handlers to the first DocWindow when it opens.
   RegAllDW ThisDocument
End Sub

' OpenWindow event handler to bind the focus event handlers.
' to each additional DocWindow when it opens and put up the menu.
Sub OpenWindow(source As Document)
   Dim dw As DocWindow
   Dim s$
   ' Since we are responding to Openwindow, windowsCount is at least 1.
   windowsCount = ThisDocument.DocWindows.Count
   ' Find the window just added.
   Set dw = ThisDocument.DocWindows(windowsCount - 1)
   ' For a list, use the name of the window as a Tag (identifying index).
   s$ = dw.Name
   ' Set the new list element with the new tag.
   Set multiDocWindows(s$) = dw
   ' Register for the Focus events.
   ' RegAllDW will not re-register windows already registered.
   RegAllDW ThisDocument
   ' Add menu 1 to the menu bar in the third-from-last position.
   AddAMenu
End Sub

' This handler is called whenever a window on the document is closed.
' We are not told which one. Find out and remove it from the list.
Sub Closewindow(source As Document)
   Dim s$
   Dim tag$ List
   ' First, make a list of tags of existing windows.
```

```
   Forall dw In source.DocWindows
      tag$(dw.Name) = dw.Name
   End Forall
   ' Then go through the stored list of windows.
   Forall dw In multiDocWindows
      ' Get the tag of this window.
      s$ = ListTag(dw)
      ' If a window's tag isn't in the tag list, then it was closed.
      If Not Iselement(tag$(s$)) Then
         ' The registration was removed when the window was closed,
         ' but the window reference is still in this list.
         ' So remove the registration flag element from the list.
         Erase regFlags(s$)
         ' And remove the window, too.
         Erase multiDocWindows(s$)
      End If
   End Forall
End Sub

' Restore the menu bar when the document closes,
' using the PreClose and PostClose events.
Function PreClose(source As Document, p1 As Variant) As Variant
   If isMenu1Up = False Then
      ' Continue the close. The menu bar has already been reset.
      PreClose = $Continue
   Else
      ' Block the close and raise the PostClose event.
      PreClose = $Block
   End If
End Function
Sub PostClose(source As Document, p1 As Variant)
' PostClose is raised only when PreClose returns $Block.
   ' Reset the menu bar.
   RemoveAMenu
   ' Now actually close the document.
   source.Close
End Sub
```

```
' Example: StartPrint, EndPrint, and CancelPrint event handlers
' The following example sends mail to someone when the document is
' successfully printed. If printing is canceled, no mail is sent.

Dim isPrintingCanceled As Variant ' Added to Global Declarations.

Sub StartPrint(source As Application)
    isPrintingCanceled = False
End Sub

Sub CancelPrint(source As Application)
   isPrintingCanceled = True
End Sub

Sub EndPrint(source As Application)
   If isPrintingCanceled = False Then
      CurrentApplication.SendMail "Joseph Folk",, _
      "Report completed and printed", "It's waiting in your printer."
   End If
End Sub
```

1-2-3: Initialize sub

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DATALINK_CLASS;H_123_DOCUMENT_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_INITIALIZE_EVENT_EXSCRIPT',1)} See example

In 1-2-3, an object's Initialize sub runs when the first non-empty event script on the object is run. Initialize doesn't run again until after the object's scripts have been unloaded, or the file has been unloaded.

Internal syntax

Initialize

Parameters

None

Usage

The Initialize sub resembles a 1-2-3 event handler, but doesn't have any parameters.

You can use Initialize to set up an object (such as declaring or initializing global variables) when the first event on it is raised and handled.

Note that you can't bind any handler scripts to events on the object in the Initialize sub using an On Event statement, unless you have a handler script attached to another event on the object by the file, because otherwise Initialize won't run.

```
' Example: Initialize sub
' Create a custom menu for later display in this document.

' In the globals for this module, declare the flags, menu, and doc window.
Dim isMenu1Up As Variant               ' Flag to prevent adding menu1 twice
Dim isMenu1Made As Variant             ' Flag to prevent creating menu1 twice
Dim menu1 As Menu                      ' Custom menu to add to the menu bar
Dim settingValue As Variant            ' Boolean property set by the menu item
Dim settingPos As Integer              ' Menu position of the property item

' In the Initialize sub on the file, build the custom Menu object.
Sub Initialize
   ' Create the custom menu.
   Set menu1 = CurrentApplication.NewMenu
   ' Set up display text for the custom menu.
   menu1.MenuText = "&Autotasks"
   menu1.MenuPrompt = "Custom automated tasks"
   ' Add items to the menu here ...
   menu1.AddItem 1, "Menu pick 1", "Prompt 1", ThisDocument, "itemHandler1"
   ' Initialize the item settings.
   settingValue = False
   settingPos = 1
   ' Initialize the flags that track menu creation and display.
   isMenu1Up = False
   isMenu1Made = True
End Sub
```

1-2-3: LostFocus event

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_LOSTFOCUS_EVENT_EXSCRIPT;H_123_GETFOCUS_EVENT_EXSCRIPT',1)} See example

Occurs when the window loses focus.

Internal syntax

LostFocus(*source*)

Parameters

source

The window object that lost focus.

Usage

You can use this event on a DocWindow object to remove handlers that are specific to the document.

A window that loses the focus is no longer the active window.

{button ,AL(`H_123_GETFOCUS_EVENT_MEMDEF;H_123_DESELECTED_EVENT_MEMDEF;H_123_SELECTED_EVENT_MEMDEF',0)} See related topics

```
' Example: GetFocus, LostFocus, and Opened event handlers
' These examples work only when there is no more than one document window open.

' Example 1
' Set a flag for disposal of resources that are only needed when
' a document (with only one document window) has focus.
' To dispose of resources when the document closes,
' use the PreClose and PostClose event handlers.
' See the PreClose event example for details.

' Global declarations.
Dim resource1Keep As Variant     ' Flag indicating that the resource should stay
available

' Set a flag to indicate that a resource is needed because the window got focus.
Sub GetFocus(source As DocWindow)
   resource1Keep = True
End Sub

' Set a flag to indicate that a resource is not needed because of the lost focus.
Sub LostFocus(source As DocWindow)
   resource1Keep = False
End Sub

' Bind the focus event handlers in the first DocWindow when it opens.
Sub Opened(source As Document)
   Set docWindow1 = ThisDocument.DocWindows(0)
   On Event GetFocus From docWindow1 Call GetFocus
   On Event LostFocus From docWindow1 Call LostFocus
End Sub

' Example 2
' Display a custom menu when the document gets focus and
' remove it when the document loses focus.
' Note 1: This example has over fifty lines of code.
' Note 2: You also need to take the custom menu down
' when the document closes. See the example for the PreClose
' and PostClose event handlers for details on how to do this.

' Global declarations.
Dim docWindow1 As DocWindow      ' Doc window for the event
Dim isMenu1Up As Variant         ' Flag to prevent adding menu1 twice
Dim menu1 As Menu                ' Custom menu to add to the menu bar
Dim settingValue As Variant      ' Boolean property set by the menu item
Dim settingPos As Integer        ' Menu position of the property item

' GetFocus event handler that displays a custom menu when the doc window gets the
focus.
Sub GetFocus(source As DocWindow)
   If isMenu1Up = False Then
      ' Add the custom menu to the menu bar.
      CurrentApplication.CurrentMenuBar.AddMenu -3, menu1
```

```
      ' Set the flag indicating that this menu is up.
      isMenu1Up = True
   End If
End Sub

' LostFocus event handler that deletes a custom menu when the doc window loses the
focus.
Sub LostFocus(source As DocWindow)
   If isMenu1Up = True Then
      CurrentApplication.ResetMenuBar
      isMenu1Up = False
   End If
End Sub


' Bind the following handler to the Opened event on the Document object.
Sub Opened(source As Document)
   ' Create the custom menu.
   Set menu1 = CurrentApplication.NewMenu
   ' Set up display text for the custom menu.
   menu1.MenuText = "&Autotasks"
   menu1.MenuPrompt = "Custom automated tasks"
   ' Add items to the menu here ...
   menu1.AddItem 1, "Menu pick 1", "Prompt 1", ThisDocument, "itemHandler1"
   ' Initialize the item settings.
   settingValue = False
   settingPos = 1
   ' Add the custom menu to the menu bar.
   CurrentApplication.CurrentMenuBar.AddMenu -3, menu1
   ' Set the flag indicating that this menu is up.
   isMenu1Up = True
   ' Bind the focus event handlers in the first DocWindow when it opens.
   Set docWindow1 = ThisDocument.DocWindows(0)
   On Event GetFocus From docWindow1 Call GetFocus
   On Event LostFocus From docWindow1 Call LostFocus
End Sub


' Global handler script for the custom menu item.
Sub itemHandler1
   ' Set the property.
   settingValue = Not settingValue
   ' Set the menu item check mark.
   If settingValue = True  Then menu1.CheckItem   settingPos
   If settingValue = False Then menu1.UncheckItem settingPos
End Sub
```

```
' Example: MenuBarReset event handler
' Set a flag so that other 1-2-3 files can put up their custom menus when they get
focus.

' In the 1-2-3 script module globals, declare the flag.
Dim isMenuBarDefault As Variant

Sub MenuBarReset(source As Application)
   ' Flag the menu bar reset.
   isMenuBarDefault = True
End Sub
```

1-2-3: MethodInvoked event

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_PLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_RANGESELECTOR_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_METHODINVOKED_EVENT_EXSCRIPT',1)} See example

Occurs when any method is called on the object.

Internal syntax

MethodInvoked(*source, name, arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9, arg10, arg11, arg12, arg13, arg14, arg15*)

Parameters

source

The object on which the method executed.

name

String. The name of the method that was called.

arg1, arg2, . . . , arg15

Variant. The values of the first 15 arguments that were passed to the method.

{button ,AL(`H_123_PROPERTYSET_EVENT_MEMDEF',0)} See related topics

```
' Example: MethodInvoked, PostSave, and PreClose event handlers
' This MethodInvoked handler sets a flag indicating that there are
' key dirty objects, and adds this object to a list of key object types
' that had methods invoked on them.
' You could use the list to formulate detailed criteria for saving a dirty file.
' (This is not done here.)
' To use this handler, bind the MethodInvoked handler in each object
' that you want to monitor, substituting the object's class name
' for BaseObject in the handler definition.  Also, bind the PreClose handler
' and initialize the flag in the document.

' In the Globals, declare the flag and list.
Public dirtyObjectList List As Variant
Public dirtyObjects As Variant
' Include LSCONST.LSS in your module, for the MessageBox.

' Bind the following handler to the MethodInvoked event on the object.
Sub MethodInvoked(source As BaseObject, P1 As String)
   ' Add the object class to the list, and set the flag.
   dirtyObjectList(Typename(source)) = True
   dirtyObjects = True
End Sub

' Before a close, put up a warning dialog box if key objects have changed.
Function PreClose(source As Document, saveChanges As Variant) As Variant
   If dirtyObjects = True Then
      MessageBox |This file contains key objects that have changed.
To preserve them, you need to save this file before closing it.|, _
      MB_OK + MB_ICONEXCLAMATION, "Pre-Close Alert"
   End If
End Function

' Reset dirtyObjects when the Document object is created and when it is saved.
Sub Opened(source As Document)
   dirtyObjects = False
End Sub
Sub PostSave(source As Document, status As Variant)
   If status = $Continue Then dirtyObjects = False
End Sub
Sub PostSaveAs(source As Document, status As Variant)
   If status = $Continue Then dirtyObjects = False
End Sub
```

1-2-3: Moved event

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_MOVED_EVENT_EXSCRIPT',1)} See example

Occurs when the window moves.

Internal syntax

Moved(*source*)

Parameters

source

The window object that moved to a new position.

```
' Example: ApplicationWindow, Left, Width, Top, Height,
'          and DocWindows properties; Moved event handler;
' Resize the window if necessary to keep it within the application window client area.

' Global declarations
Dim docWindow1 As DocWindow      ' The doc window whose Moved event is handled.

' Handler for the Moved event.
Sub winMoved(source As DocWindow)
   Dim appWindow As ApplicationWindow
   Set appWindow = CurrentApplication.ApplicationWindow
   If source.Left + source.Width > appWindow.Width Then
      source.Width = appWindow.Width - source.Left
   End If
   If source.Top + source.Height > appWindow.Height Then
      source.Height = appWindow.Height - source.Top
   End If
End Sub

' Bind the winMoved handler in the DocumentOpened event.
Sub DocumentOpened(source As Application)
   Set docWindow1 = ThisDocument.DocWindows(0)
   On Event Moved From docWindow1 Call winMoved
End Sub

' If you want to do the resize in multiple document windows,
' you need to bind the handler for those document windows in the OpenWindow event.
```

1-2-3: MenuBarReset event

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_MENUBARRESET_EVENT_EXSCRIPT',1)} See example

Occurs when the ResetMenuBar method executes.

Internal syntax

MenuBarReset(*source*)

Parameters

source

Application. The 1-2-3 application in which the menu bar was reset.

{button ,AL(`H_123_CLOSEWINDOW_EVENT_MEMDEF;H_123_DOCUMENTOPENED_EVENT_MEMDEF;H_123_OPENED_EVENT_MEMDEF;H_123_LOSTFOCUS_EVENT_MEMDEF;H_123_RESETMENUBAR_METHOD_MEMDEF',0)} See related topics

1-2-3: NameChange event

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_PLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_NAMECHANGE_EVENT_EXSCRIPT',1)} See example

Occurs when an object's name has been changed.

Internal syntax

NameChange(*source*)

Parameters

source

The object whose name changed.

```
' Example: NameChange event handler
' Check the range whose name changed to see if its old name is needed.

' Declare the needed range.
Dim neededRange As Range
' Sub to set up the range whose name is to be monitored.
Sub setNeededRange
   Set neededRange = Bind("C10..C11")
   neededRange.Name = "Init Name"
   ' Bind the handler sub to the event on this range.
   On Event NameChange From neededRange Call nameChangeHandler
End Sub
' NameChange event handler.
Sub nameChangeHandler(source As Range)
   If source.CoordinateString = neededRange.CoordinateString Then
      MessageBox |Warning: The range | & source.CoordinateString & | has been renamed.
         The scripts using the old name need to be updated.|, MB_OK + MB_ICONQUESTION,
_
         "Script 1"
   End If
End Sub
' Sub for testing the event handler.  This sub will raise the event.
Sub setNeededRangeName
   neededRange.Name = "New Name"
End Sub
```

1-2-3: Opened event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENED_EVENT_EXSCRIPT',1)} See example

Occurs when the document opens.

Internal Syntax

Opened(*source*)

Parameters

source

Document. The object on which the event was raised.

```
' Example: Opened event handler
' The following example adds a custom menu to a document whenever the

' document opens.
' Note: This example only adds a menu.
' To remove the menu when the document closes, use Example #1
' for the PostClose event.

Dim myMenu As Menu     ' Add to Global declarations

Sub Opened(Source As Document)
   ' Create a menu.
   Set myMenu = CurrentApplication.NewMenu
   myMenu.MenuText = "Menu title"
   ' Script to set up the menu's items omitted.
   ' Place the menu at the 5th position in the menu bar.
   CurrentApplication.CurrentMenubar.AddMenu 5, myMenu
End Sub
```

1-2-3: OpenWindow event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENWINDOW_EVENT_EXSCRIPT',1)} See example

Occurs when a workbook (document) window opens.

Internal Syntax

OpenWindow(*source*)

Parameters

source

Document. The object on which the event occurred.

```
' Example: OpenWindow event handler
' The following example sets a global variable when the document is opened
' in a window.

Dim windowOpened As Variant ' A Global Declaration

Sub Openwindow(Source As Document)
    windowOpened = True
End Sub
```

1-2-3: Poll1 event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_POLL1_POLL2_POLL3_POLL4_EVENT_EXSCRIPT',1)} See example

This event occurs at the interval specified in the StartPoll method that activated it.

Internal Syntax

Poll1(*source*)

Parameters

source

Document. The object on which the event occurred.

Usage

Each document has four poll events. These can be used in any combination and can trigger at overlapping intervals. For example, one poll event could save the document at one hour intervals, another poll event could print the document at two hour intervals, and so on.

{button ,AL(`H_123_STARTPOLL_METHOD_MEMDEF;H_123_POLL2_EVENT_MEMDEF;H_123_POLL3_EVENT_MEMDEF;H_123_POLL4_EVENT_MEMDEF',0)} See related topics

```
' Example: Poll1, Poll2, Poll3, Poll4 event handler


' Like all poll events, the following example for the Poll1
' event handler can be used for any of the other polls
' (Poll2, Poll3, or Poll4). For example, to use this example
' for Poll 2, change the two instances of 1 to 2.


' In the following example, the script for the Opened event
' turns on Poll1 and specifies that it should raise
' at one hour intervals (every 3600000 milliseconds)
' until the document closes. Then, the script for the Poll1
' event saves the document once per hour.

Sub Opened(Source As Document)
   CurrentDocument.StartPoll 1, 3600000, 0
End Sub

Sub Poll1(Source As Document)
   Source.Save
End Sub
```

1-2-3: Poll2 event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_POLL1_POLL2_POLL3_POLL4_EVENT_EXSCRIPT',1)} See example

This event occurs at the interval specified in the StartPoll method that activated it.

Internal Syntax

Poll2(*source*)

Parameters

source

Document. The object on which the event occurred.

Usage

Each document has four poll events. These can be used in any combination and can trigger at overlapping intervals. For example, one poll event could save the document at one hour intervals, another poll event could print the document at two hour intervals, and so on.

{button ,AL(`H_123_STARTPOLL_METHOD_MEMDEF;H_123_POLL1_EVENT_MEMDEF;H_123_POLL3_EVENT_MEMDEF;H_123_POLL4_EVENT_MEMDEF',0)} See related topics

1-2-3: Poll3 event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_POLL1_POLL2_POLL3_POLL4_EVENT_EXSCRIPT',1)} See example

This event occurs at the interval specified in the StartPoll method that activated it.

Internal Syntax

Poll3(*source*)

Parameters

source

Document. The object on which the event occurred.

Usage

Each document has four poll events. These can be used in any combination and can trigger at overlapping intervals. For example, one poll event could save the document at one hour intervals, another poll event could print the document at two hour intervals, and so on.

{button ,AL(`H_123_STARTPOLL_METHOD_MEMDEF;H_123_POLL2_EVENT_MEMDEF;H_123_POLL1_EVENT_MEMDEF;H_123_POLL4_EVENT_MEMDEF',0)} See related topics

1-2-3: Poll4 event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_POLL1_POLL2_POLL3_POLL4_EVENT_EXSCRIPT',1)} See example

This event occurs at the interval specified in the StartPoll method that activated it.

Internal Syntax

Poll4(*source*)

Parameters

source

Document. The object on which the event occurred.

Usage

Each document has four poll events. These can be used in any combination and can trigger at overlapping intervals. For example, one poll event could save the document at one hour intervals, another poll event could print the document at two hour intervals, and so on.

{button ,AL(`H_123_STARTPOLL_METHOD_MEMDEF;H_123_POLL2_EVENT_MEMDEF;H_123_POLL3_EVENT_MEMDEF;H_123_POLL1_EVENT_MEMDEF',0)} See related topics

1-2-3: PostClose event

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_PRECLOSE_EVENT_EXSCRIPT',1)} See example

Occurs only after the PreClose event event has blocked a close. If the PreClose event allows the object to close (without blocking the close), the script attached to the PostClose event never runs, because the object has closed.

Internal Syntax

PostClose(*source, status*)

Parameters

source

The object on which the event occurred.

status

Variant (enumeration). Indicates that the document close was blocked. The value for this argument is always $Block.

1-2-3: PostSaveAs event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_PRESAVEAS_EVENT_EXSCRIPT',1)} See example

The PostSaveAs event is raised after a SaveAs command has either occurred or been blocked.

Internal Syntax

PostSaveAs(*source, status*)

Parameters

source

Document. The object on which the event occurred.

status

Variant (enumeration). Indicates whether the SaveAs command is blocked or not. The following table lists the possible values for the status argument.

Value	Description
$Block	The SaveAs command is blocked.
$Continue	The SaveAs command is not blocked.

{button ,AL(`H_123_PRESAVEAS_EVENT_MEMDEF',0)} See related topics

1-2-3: PostSave event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_PRESAVE_EVENT_EXSCRIPT',1)} See example

The PostSave event is raised after a Save command has either occurred or been blocked.

Internal Syntax

PostSave(*source, status*)

Parameters

source

Document. The object on which the event occurred.

status

Variant (enumeration). Indicates whether the Save command is blocked or not. The following table lists the possible values for the status argument.

Value	Description
$Block	The Save command is blocked.
$Continue	The Save command is not blocked.

1-2-3: PreClose event

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_PRECLOSE_EVENT_EXSCRIPT',1)} See example

Occurs when an object is about to close. The script you add to this event should determine whether to allow or block a document close.

The script attached to this event can get properties, but it cannot set properties or call object methods.

Internal Syntax

PreClose(*source, savechanges*)

Parameters

source

The object on which the event occurred.

savechanges

Variant (Boolean). Indicates whether to save any changes that have been made to the file (value True) or not (value False).

Return values

The following table lists the possible return values for this event.

Value	Description
$Block	Blocks the close and raises the PostClose event.
$Continue	Continues the close and doesn't raise the PostClose event.

Usage

You can use PreClose handlers when you want your script to do something whenever the object closes.

```
' Example: ResetMenuBar method; PreClose and PostClose event handlers
' Example 1

' In the following example, the PreClose and PostClose events are used to
' reset the menu bar whenever the document is closed.
' This removes all custom menus displayed while the document was open.

' Global declaration.
Dim isMenu1Up As Variant   ' Boolean flag indicating a custom menu is up.

Function PreClose(source As Document, p1 As Variant) As Variant
   If isMenu1Up = False Then
      ' Continue the close. The menu bar has already been reset.
      PreClose = $Continue
   Else
      ' Block the close and raise the PostClose event.
      PreClose = $Block
   End If
End Function

Sub PostClose(source As Document, p1 As Variant)
' PostClose is raised only when PreClose returns $Block.
   ' Reset the menu bar.
   CurrentApplication.ResetMenuBar
   ' Allow the Close method to proceed.
   isMenu1Up = False
   ' Now actually close the document.
   source.Close
End Sub

' Example 2
' The following PreClose event displays a message box which asks whether the
' user really wants to close the document. If the user clicks YES, the
' PostClose event is never raised.

Function PreClose(Source As Document, P1 As Variant) As Variant
    r = Msgbox("Close this document?", 4, "Confirmation")
    If r = 6 Then
        ' User clicked YES.
        ' Continue the close.
        PreClose = $Continue
    Else
        ' User clicked NO.
        ' Block the close and raise PostClose.
        PreClose = $Block
    End If
End Function

' PostClose is raised only when PreClose returns $Block.
Sub PostClose(Source As Document, P1 As Variant)
    Msgbox "Closure has been blocked!",,"1-2-3"
End Sub
```

1-2-3: PreSaveAs event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_PRESAVEAS_EVENT_EXSCRIPT',1)} See example

Occurs when a document is about to be saved using the SaveAs command. The script you add to this event should determine whether to allow or block the SaveAs command.

The script attached to this event can get properties, but it cannot set properties or call object methods.

Internal Syntax

PreSaveAs(*source, filename,* [*location*], [*filetype*], [*backup*], [*password*], [*addtorecentfiles*])

Parameters

source

Document. The object on which the event occurred.

filename

String. The name of the file, or the name and full path to the file.

location

(Optional) String. The container or path where the file is to be saved.

filetype

(Optional) String. The file format to be used. The default is 1-2-3 97 workbook format. The following table lists the allowed values for this parameter.

Value	Description
"1-2-3 (123)"	1-2-3 97 workbook
"1-2-3 (WK4)"	1-2-3 for Windows Release 4
"1-2-3 (WK3)"	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
"1-2-3 (WK1)"	1-2-3 for DOS Release 2
"SmartMaster (12M)"	1-2-3 SmartMaster template
"Text (TXT)"	Text
"Comma-delimited text (TXT)"	Comma-delimited text
"Excel Worksheet (XLS)"	Microsoft Excel worksheet
"Excel Workbook (XLW)"	Microsoft Excel workbook

backup

(Optional) Variant (Boolean). Specifies whether to back up the file (value True) or not (value False). If this argument is specified as False, 1-2-3 replaces the file with the one being saved.

password

(Optional) String. A password associated with the document. If this argument is not specified, the file has no password.

addtorecentfiles

(Optional) Variant (Boolean). Specifies whether to add the file to the most recent file list (value True) or not (value False). The default is not to add it (False).

Return values

The following table lists the possible return values for this event.

Value	Description
$Block	Blocks the close and raises the PostSaveAs event.
$Continue	Continues the close and doesn't raise the PostSaveAs event.

{button ,AL(`H_123_POSTSAVEAS_EVENT_MEMDEF',0)} See related topics

```
' Exmple: PreSaveAs and PostSaveAs event handlers
' Allow the user to block a SaveAs command or not to.

Function PreSaveAs(Source As Document, P1 As String, P2 As String, P3 As String, P4 As
Variant, P5 As String, P6 As Variant) As Variant
    r = Msgbox("Save this document?", 4, "Confirmation")
    If r = 6 Then
        ' User clicked YES.
        ' Continue the SaveAs, then raise the PostSaveAs event.
        PreSaveAs = $Continue
    Else
        ' User clicked NO.
        ' Block the SaveAs and raise the PostSaveAs event.
        PreSaveAs = $Block
    End If
End Function

Sub PostSaveAs(Source As Document, P1 As Variant)
   ' P1 will be 1 if the SaveAs command was blocked
   ' or 0 if it wasn't.
   If (P1 = $Block) Then
      Msgbox "This SaveAs command was blocked.",, "1-2-3"
   End If
End Sub
```

1-2-3: PreSave event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_PRESAVE_EVENT_EXSCRIPT',1)} See example

Occurs when a document is about to be saved. The script you add to this event should determine whether to allow or block a save.

The script attached to this event can get properties, but it cannot set properties or call object methods.

Internal Syntax

PreSave(*source*)

Parameters

source

Document. The object on which the event occurred.

```
' Example: PreSave and PostSave event handlers
' The following PreSave event handler demonstrates that
' you can block a document save. Then the PostSave event handler
' indicates whether the document was saved.

' Make a document's PreSave event handler as follows:
Function PreSave(Source As Document) As Variant
     r = Msgbox("Save this document?", 4, "Confirmation")
     If r = 6 Then
          ' User clicked YES.
          ' Continue the Save, then raise PostSave.
          PreSave = $Continue
     Else
          ' User clicked NO.
          ' Block the Save and raise PostSave.
          PreSave = $Block
     End If
End Function

' Make the same document's PostSave event handler as follows:
Sub PostSave(Source As Document, P1 As Variant)
     ' P1 will be $Block if the Save was blocked or $Continue if it wasn't.
     If (P1 = $Block) Then
          Msgbox "This Save was blocked.",, "1-2-3"
     Else
          Msgbox "Document saved.",, "1-2-3"
     End If
End Sub
```

1-2-3: PropertySet event

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_PLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_PROPERTYSET_EVENT_EXSCRIPT',1)} See example

Occurs when any property is set on the object.

Internal syntax

PropertySet(*source, name, value*)

Parameters

source

The object on which the property was set.

name

String. The name of the property that was set.

value

Variant. The new value of the property.

{button ,AL(`H_123_METHODINVOKED_EVENT_MEMDEF',0)} See related topics

```
' Example: FormatName property; PropertySet event handler
' Display a warning if a range's format is changed to a problematic format.
Sub PropertySet(source As Range, sourcename As String, value As Variant)
   If sourcename = "FormatName" And source.FormatName <> "General" Then
      MessageBox "Range format changed from General to " & value & _
         ". Some values may not display.", MB_OK + MB_ICONINFORMATION, _
         "Script 1"
   End If
End Sub
```

1-2-3: Resized event

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_RESIZED_EVENT_EXSCRIPT',1)} See example

Occurs when a window has been resized.

Internal syntax

Resized(*source*)

Parameters

source

The window that was resized.

{button ,AL(`H_123_MOVED_EVENT_MEMDEF',0)} See related topics

```
' Example: Resized event handler
' Resize a control when the parent window Resized event occurs.

' Global declarations.
Dim scale As Single             ' Scale factor for resizing
Dim docWindow1 As DocWindow     ' Doc window in which to resize

' DocumentOpened event handler to bind the Resized event handler.
Sub DocumentOpened(source As Application)
   Set docWindow1 = ThisDocument.DocWindows(0)
   ' Bind the Resized event handler in the first document window.
   On Event Resized From docWindow1 Call myResizedHandler
   ' Use a scale factor of 0.5 and convert to twips (1/1440 inch) from pixels
   ' for a monitor with a screen resolution of 120 dots per inch.
   ' DrawObjects are in twips, but window size is in pixels.
   ' Get the screen DPI for Windows 95 from HKEY_CURRENT_CONFIG\Display\Settings.
   scale = 0.5 * 1440 / 120
End Sub

' Handler sub for the Resized event.
Sub myResizedHandler(source As DocWindow)
   ' Resize the first DrawObject.
   ThisDocument.DrawnObjects(0).Height = scale * source.Height
   ThisDocument.DrawnObjects(0).Width = scale * source.Width
End Sub

' The above handler is satisfactory for centrally located objects,
' but you may want a more precise size adjustment for offset objects,
' based on the position of the object.
```

1-2-3: Selected event

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_QUERY_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPTITLE_CLASS;H_123_PLOT_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_SELECTED_EVENT_EXSCRIPT',1)} See example

Occurs when the object is selected.

Internal syntax

Selected(*source*)

Parameters

source

The object that was selected.

{button ,AL(`H_123_DESELECTED_EVENT_MEMDEF;H_123_CLICK_EVENT_MEMDEF;H_123_GETFOCUS_EVENT_MEMDEF;H_123_LOSTFOCUS_EVENT_MEMDEF',0)} See related topics

```
' Example: Cells and OpenDocumentFromInternet methods;
'          Selected event handler
' Launch a web browser using a URL entered in the source cell.
Sub Selected(source As Range)
   Dim sourceURL As String
   ' Take the URL from the first cell in the source range.
   sourceURL = source.Cells(0).Contents
   ' Delete the label-prefix character that 1-2-3 puts in label cells.
   sourceURL = Mid(sourceURL, 2)
   ' Verify that the selection was intentional.
   If IDYES = MessageBox("Browse the page " & sourceURL & " ?", _
                         MB_YESNO + MB_ICONQUESTION, _
                         ThisDocument.Name & " script") Then
   ' Launch the Web browser.
   Dim taskID As Integer
   taskID = Shell(|C:\Program Files\Netscape\Navigator\Program\netscape | & sourceURL,
1)
   End If
   ' Note: It's not necessary to assume the user has
   ' Netscape Navigator installed in a particular path.
   ' You can find the user's default browser using the script
   ' for the Internet Tool "Create a button with a link to a URL."
   ' Click View - Show Internet Tools, create the button,
   ' and copy its script.
End Sub
```

1-2-3: SheetChange event

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHEETCHANGE_EVENT_EXSCRIPT',1)} See example

Occurs when a different sheet becomes the current sheet.

Internal Syntax

SheetChange(*source*)

Parameters

source

Document. The object on which the event occurred.

```
' Example: SheetChange event handler
' The SheetChange event is useful for controlling whether the sheet should
' really change.

' Assume there are two sheets and each one contains a button. You want to
' prevent navigation from one sheet to another until a button is clicked.

' Global declarations.
Dim LastSheet As Variant
Dim AllowSheetChange As Integer

' Set up each button's Click event.
Sub Click(Source As Buttoncontrol)
    AllowSheetChange = True
End Sub

' Set up the SheetChange event.
Sub SheetChange(Source As Document)
    If AllowSheetChange = False Then
        AllowSheetChange = True
        CurrentDocument.Sheets(LastSheet.SheetName).TurnTo
    Else
        Set LastSheet = CurrentDocument.CurrentSheet
        AllowSheetChange = False
    End If
End Sub

' Set up the document's Opened event.
Sub Opened(Source As Document)
    Set LastSheet = CurrentDocument.CurrentSheet
    AllowSheetChange = False
End Sub
```

1-2-3: StartPrint event

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTPRINT_ENDPRINT_CANCELPRINT_EVENT_EXSCRIPT',1)} See example

Occurs after a user clicks Print in the Print dialog box or calls the Print method, and 1-2-3 has started transmitting print pages to the print queue.

Internal Syntax

StartPrint(*source*)

Parameters

source

Application. The object on which the event occurred.

1-2-3: Terminate sub

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DATALINK_CLASS;H_123_DOCUMENT_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_TERMINATE_EVENT_EXSCRIPT',1)} See example

In 1-2-3, an object's Terminate sub runs when the object is deleted.

Internal syntax

Terminate

Parameters

None

Usage

The Terminate sub resembles a 1-2-3 event, but doesn't have any parameters.

You can use Terminate to clean up after an object (such as closing files and deleting unneeded auxiliary objects) when the object is deleted.

```
' Example: Terminate sub
' Delete a custom object not needed outside this object.

' In the Initialize sub, instantiate the custom object.
Sub Initialize
Dim myInstance As New myClass
End Sub

' In the Terminate sub, delete the object reference.
Sub Terminate
   Delete myInstance
End Sub
```

1-2-3: ValueChange event

{button ,AL(`H_123_BUTTONCONTROL_CLASS;H_123_EDITTEXT_CLASS;H_123_MAPTITLE_CLASS',0)} See list of classes

{button ,AL(`H_123_VALUECHANGE_EVENT_EXSCRIPT',1)} See example

Occurs when the Text property of the object is changed.

Internal syntax

ValueChange(*source*)

Parameters

source

The object on which the event occurred.

```
' Example: ValueChange event handler
' Copy the new Text value into a cell.
Sub ValueChange (source As EditText)
   [A:C10].Contents = source.Text
End Sub
```

1-2-3 predefined global product variables

1-2-3 supports five predefined global variables that greatly simplify script object references.

Variable	Description
CurrentApplication	Represents the current session of 1-2-3 and is the object at the top of the entire hierarchy. Uses the properties and methods of the Application class.
CurrentDocument	Represents the current 1-2-3 document (.123 file) in the current 1-2-3 session. Uses the properties and methods of the Document class.
CurrentWindow	Represents the window in which 1-2-3 displays the current document and uses the properties and methods of the DocWindow class.
Selection	Represents the currently selected object, for example, the currently selected range, chart, or graphic object.
ThisDocument	Represents the document that contains the script that is currently running. Uses the properties and methods of the Document class.

1-2-3 LotusScript constants

You can select the following 1-2-3 constants from the Browser:

Constant	Description
ClearBorder	Clear the cell border
ClearComments	Clear the cell comment
ClearData	Clear the cell contents
ClearFormat	Clear the data format
ClearScripts	Clear the scripts
ClearStyle	Clear the style
MemberOfCollection	The range is a member of the current worksheet collection
MultiCellOverlapped	The range contains versions, and some cells overlap another range that contains versions
NotVersioned	The range contains no versions
PasteBorders	Paste object borders
PasteComments	Paste cell comments
PasteData	Paste cell contents, but leave the styles in the target object intact
PasteFormulas	Paste both cell contents and styles, but convert all formulas to values
PasteDrawObjectsWithRange	Paste any graphic objects associated with the 1-2-3 range on the Clipboard
PasteStyleAndNumberFormats	Paste all formatting done with the InfoBox
SingleCellOverlapped	The range contains versions, and one cell overlaps another range that contains versions
VersionBordered	The range contains versions, and the version border is visible
Versioned	The range contains versions
VersionedHiddenCurrent	The range contains versions, and they are hidden

1-2-3: Activate method

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_ACTIVATE_METHOD_EXSCRIPT',1)} See example

Activates the document, application, or window.

Syntax

*object.***Activate**

Parameters

None

Return values

None

Usage

For a Document object, this method activates the last active DocWindow it finds.

{button ,AL(`H_123_GOTO_METHOD_MEMDEF;H_123_SCROLLTOACTIVECELL_METHOD_MEMDEF;H_123_NEWDOCUMENT_METHOD_MEMDEF;H_123_ACTIVE_PROPERTY_MEMDEF;H_123_ACTIVEDOCUMENT_PROPERTY_MEMDEF;H_123_ACTIVEDOCWINDOW_PROPERTY_MEMDEF',0)} See related topics

```
' Example: Activate method and NewDocument method
' Create 2 documents and activate the first one.
Dim document1 As Document, document2 As Document
Set document1 = CurrentApplication.NewDocument("Doc #1")
' Doc #1 is now current.
Set document2 = CurrentApplication.NewDocument("Doc #2")
' Doc #2 is now current.
' To make Doc #1 current, activate it.
document1.Activate
```

1-2-3: AddItem method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS;',0)} See list of classes

{button ,AL(`H_123_ADDITEM_METHOD_EXSCRIPT',1)} See example

Adds a non-submenu item to a menu or menu bar and binds a specified script to it. The item script is executed when the user chooses the item.

Syntax

object.**AddItem** *position, menutext, menuprompt, scriptdocument, scriptname*

Parameters

position

Long. The new item's position in the menu.

Value	Description
Positive integer	The new item's position in the menu, counting forward from the beginning. The value 1 means after the first existing item. If *position* exceeds the position of the last existing item, the new item is placed at the end.
Zero	Place the new item before the first existing menu item.
Negative integer	The new item's position in the menu, counting backward from the end. The value –1 means the last position.

menutext

String. Text for the new item to display in the menu. An & (ampersand) before any letter specifies the shortcut key for the item.

menuprompt

String. The long prompt for the item.

scriptdocument

Document. The file containing the global script to be called by the item. This must be either a reference to a Document object in the form [*docname*.123] or the predefined global variable ThisDocument. For example, [mysheet.123].

scriptname

String. The name of the global script or sub to call when the user chooses the menu item. This script can employ the full set of 1-2-3 methods and properties, as well as LotusScript language elements.

Return values

None

Usage

An item added by this method only executes its script, and doesn't display a submenu. To add a submenu to a menu or menu bar, use the AddMenu method.

If the item script sets a property, you can indicate the status of the setting by calling the CheckItem or UncheckItem method in the item script.

You can't add an item or menu between any special menus that 1-2-3 displays in specific contexts.

{button ,AL(`H_123_ADDSEPARATOR_METHOD_MEMDEF;H_123_CHECKITEM_METHOD_MEMDEF;H_123_DISABLEITEM_METHOD_MEMDEF;H_123_ENABLEITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_UNCHECKITEM_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: AddItem method and NewMenu method
' Create a menu and add 2 menu items to it.
' You need item handler scripts available for each item in the menu.
' The sample item handlers for this script just flip a Boolean property.

' Global declarations.
Dim menu1 As Menu                       ' Custom menu to add to the menu bar
Dim settingValue1 As Variant            ' The Boolean property to be set by the menu
item
Dim settingPos1 As Long                 ' The menu position of the property item
Dim settingValue2 As Variant            ' The Boolean property to be set by the menu
item
Dim settingPos2 As Long                 ' The menu position of the property item

' Create the custom menu.
Sub createMenu
   ' Create the custom menu object.
   Set menu1 = CurrentApplication.NewMenu
   ' Set up display text for the custom menu.
   menu1.MenuText = "&Autotasks"
   menu1.MenuPrompt = "Custom automated tasks"
   ' Initialize the custom menu items.
   settingPos1 = 1
   settingPos2 = 2
   settingValue1 = False
   settingValue1 = False
   ' Add the items to the custom menu.
   menu1.AddItem settingPos1, "Menu pick 1", "Prompt 1", ThisDocument, "itemHandler1"
   menu1.AddItem settingPos2, "Menu pick 2", "Prompt 2", ThisDocument, "itemHandler2"
   ' Add the custom menu to the menu bar in the third-from-last position.
   CurrentApplication.CurrentMenuBar.AddMenu -3, menu1
End Sub

' Handler script for the first custom menu item.
' It reverses a Boolean property and indicates the setting with a check mark.
Sub itemHandler1
   ' Set the property.
   settingValue1 = Not settingValue1
   ' Set the menu item check mark.
   If settingValue1 = True  Then menu1.CheckItem   settingPos1
   If settingValue1 = False Then menu1.UncheckItem settingPos1
End Sub

' Handler script for the second custom menu item.
' It reverses a Boolean property and indicates the setting with a check mark.
Sub itemHandler2
   ' Set the property.
   settingValue2 = Not settingValue2
   ' Set the menu item check mark.
   If settingValue2 = True  Then menu1.CheckItem   settingPos2
   If settingValue2 = False Then menu1.UncheckItem settingPos2
End Sub
```

1-2-3: AddMenu method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS;',0)} See list of classes

{button ,AL(`H_123_ADDMENU_METHOD_EXSCRIPT',1)} See example

Adds a submenu to a menu, or a new menu to a menu bar.

Syntax

object.**AddMenu** *position, newmenu*

Parameters

position

Long. The new menu's position in the menu.

Value	Description
Positive integer	The new menu's position in the menu, counting forward from the beginning. The value 1 means after the first existing item or menu. If *position* exceeds the last existing item's position, the new menu will be placed at the end.
Zero	Place the new menu before the first existing item or menu.
Negative integer	The new menu's position in the menu, counting backward from the end. The value –1 means the last position.

newmenu

Menu. The menu to add.

Return values

None

Usage

To add a menu to the current menu bar, run this method on the Application.CurrentMenuBar property, which contains the current menu bar object.

To add a non-submenu item to a menu, instead of adding a submenu, use the AddItem method. A non-submenu item only executes its script when clicked, and doesn't display a submenu.

1-2-3 doesn't add menus between any special menus that 1-2-3 displays in specific contexts. If *position* is between special menus, 1-2-3 adds the new menu next to an adjacent fixed menu.

For menus participating in OLE in-place editing, you must add your menu next to the fixed menu of the same type as the embedded participation type of your menu. For example, if the EmbeddedParticipation property of your menu has the value $FileGroup, you must add your menu next to the File menu.

{button ,AL(`H_123_EMBEDDEDPARTICIPATION_PROPERTY_MEMDEF;H_123_ADDITEM_METHOD_MEMDEF;H_123_ADDSEPARATOR_METHOD_MEMDEF;H_123_CHECKITEM_METHOD_MEMDEF;H_123_DISABLEITEM_METHOD_MEMDEF;H_123_ENABLEITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_REPLACEMENU_METHOD_MEMDEF;H_123_RESETMENUBAR_METHOD_MEMDEF;H_123_UNCHECKITEM_METHOD_MEMDEF;H_123_MENUBARRESET_EVENT_MEMDEF;H_123_GETMENU_METHOD_MEMDEF;H_123_GETMENUPOSITION_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_CURRENTMENUBAR_PROPERTY_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: AddMenu, NewMenu, AddItem, ResetMenuBar, and Close methods;
'          CurrentMenuBar, MenuText, and MenuPrompt properties;
'          Opened, PreClose, and PostClose events.

' Create a menu and add a menu item to it. Add the menu to the menu bar.
' For example, do this in the Document.Opened event handler.

' This example is suitable when the custom menu will not interfere
' with other documents that may be open. If you don't want the custom menu up
' when another document is current, use the example for the GetFocus method
' instead of this example.

' In the Globals, declare the submenu and item position,
' so the item handler can find them.
Dim isMenu1Up As Variant         ' Flag to prevent adding menu1 twice
Dim menu1 As Menu                ' Custom menu to add to the menu bar
Dim docWindow1 As DocWindow      ' First doc window
Dim settingValue As Variant      ' The Boolean property to be set by the menu item
Dim settingPos As Long           ' The position of the menu item in the menu

' Build and display the custom menu when the document opens.
' Bind this event handler to the Opened event on your document.
Sub Opened(Source As Document)
   ' Initialize the menu flag.
   isMenu1Up = False
   ' Create the custom menu.
   Set menu1 = CurrentApplication.NewMenu
   ' Set up display text for the custom menu.
   menu1.MenuText = "&Autotasks"
   menu1.MenuPrompt = "Custom automated tasks"
   ' Add items to the menu here ...
   menu1.AddItem 1, "Task 1", "Prompt 1", ThisDocument, "itemHandler1"
   ' Initialize the item settings.
   settingValue = False
   settingPos = 1
   ' Add menu 1 to the menu bar in the third-from-last position.
   CurrentApplication.CurrentMenuBar.AddMenu -3, menu1
   isMenu1Up = True
End Sub

' Handler for the menu item.
Sub itemHandler1
   ' Set the property.
   settingValue = Not settingValue
   ' Set the menu item check mark.
   If settingValue = True  Then menu1.CheckItem   settingPos
   If settingValue = False Then menu1.UncheckItem settingPos
End Sub

' Remove the custom menu when the document is closed.
' Bind these handlers to the PreClose and PostClose events.
```

```
Function Preclose(Source As Document, P1 As Variant) As Variant
   If isMenu1Up = False Then
      ' Continue the close. The menu bar has already been reset.
      PreClose = $Continue
   Else
      ' Block the close and raise the PostClose event.
      PreClose = $Block
   End If
End Function
Sub Postclose(Source As Document, P1 As Variant)
' PostClose is raised only when PreClose returns $Block.
   ' Reset the menu bar.
   CurrentApplication.ResetMenuBar
   ' Allow the Close method to proceed.
   isMenu1Up = False
   ' Now actually close the document.
   source.Close
End Sub
```

1-2-3: AddOverlay method

{button ,AL(`H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDOVERLAY_METHOD_EXSCRIPT',1)} See example

Adds an overlay to the map.

Syntax

map.**AddOverlay** *overlayname,* [*location*]

Parameters

overlayname

String. The file name of the overlay (for example, "USA.TV")

location

(Optional) String. The location of the overlay file.

Return values

None

Usage

You can use this method to add broader context or more detail, such as smaller divisions, to the map.

{button ,AL(`H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_123_NEWMAP_METHOD_MEMDEF',0)} See related topics

```
' Example: AddOverlay, NewMap methods
' Add an overlay USA.TV to a world map.
' First, create a map of data into colors by country on a world map.
[A:B2].Contents = "United States"
[A:B3].Contents = "Germany"
[A:B4].Contents = "Australia"
[A:C2].Contents = "20"
[A:C3].Contents = "30"
[A:C4].Contents = "40"
Dim worldMap As Map
Set worldMap = [A].NewMap(1440, 1440, 8640, 7200, [A:B2..A:C4], "World Countries")
' Add an overlay of state boundaries to the US part of the map.
worldMap.AddOverlay "usa.tv"
```

1-2-3: AddPoint method

{button ,AL(`H_123_DRAWLINE_CLASS;H_123_FREEHAND_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDPOINT_METHOD_EXSCRIPT',1)} See example

Adds a new point to a Polyline, Polygon, or Freehand drawing, and draws a line to that point.

Syntax

drawobject.**AddPoint** *xposition, yposition*

Parameters

xposition

Long. The horizontal position of the new point, relative to the origin of the spreadsheet (the upper left corner). The units are twips.

yposition

Long. The vertical position of the new point, relative to the origin of the spreadsheet. The positive direction, like row numbers, is down. The units are twips.

Return values

None

{button ,AL(`H_123_MOVEPOINT_METHOD_MEMDEF;H_123_NEWDRAWLINE_METHOD_MEMDEF;H_123_NEWFREEHAND_METHOD_MEMDEF;H_123_NEWPOLYGON_METHOD_MEMDEF;H_123_NEWPOLYLINE_METHOD_MEMDEF;H_123_POINTX_METHOD_MEMDEF;H_123_POINTY_METHOD_MEMDEF;H_123_POINTCOUNT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: AddPoint and NewPolygon methods
' Draw a triangle on sheet A pointing left.
Dim myPolygon As Polygon
' Begin by creating a Polygon object and drawing the first line segment.
Set myPolygon = [A].NewPolygon (1440, 2160, 2880, 2880)
' Draw the second line segment.
myPolygon.AddPoint 2880, 1440
' 1-2-3 automatically draws the third line segment,
' by connecting the first point to the last.
```

1-2-3: AddSelectField method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDSELECTFIELD_METHOD_EXSCRIPT',1)} See example

Adds a field to this DataQuery object. The field is displayed as the last field in the query table, unless you specify the existing field before which the new field is to be added.

Syntax

dataquery.**AddSelectField** *newfield,* [*nextfield*]

Parameters

newfield

String. The name of the field to add to the query.

nextfield

(Optional) String. The name of the field before which the new field is to be added. If you omit this parameter, the new field is added as the last field in the query.

Return values

None

{button ,AL(`H_123_CREATECOMPUTEDFIELD_METHOD_MEMDEF;H_123_DELETECOMPUTEDFIELD_METHOD_MEMDEF;H_123_FIELDAGGREGATETYPE_METHOD_MEMDEF;H_123_FIELDALIAS_METHOD_MEMDEF;H_123_FIELDEXAMPLEVALUE_METHOD_MEMDEF;H_123_REMOVESELECTFIELD_METHOD_MEMDEF;H_123_SELECTFIELDS_PROPERTY_MEMDEF;H_123_NEWQUERY_METHOD_MEMDEF',0)} See related topics

```
' Example: OutputLocation and SelectFields properties;
'          AddSelectField, DeleteQuery, NewQuery, Refresh methods
' Create a new query with four fields, then modify the query field selection.

' First, set up the query.
[A:B2].Contents = "Field 1"
[A:C2].Contents = "Field 2"
[A:D2].Contents = "Field 3"
[A:E2].Contents = "Field 4"
CurrentDocument.NewQuery "Query1", "A:B2..A:E10"
' Run the query.
[Query1].OutputLocation = "A:A13"
[Query1].Refresh

' Reduce the fields in the query to fields 1 and 2.
[Query1].SelectFields = "Field 1;Field 2"

' Add Field 4 to the end of the query.
[Query1].AddSelectField "Field 4"
' Add Field 3 to the query before Field 4.
[Query1].AddSelectField "Field 3", "Field 4"

' If necessary, delete the query when finished.
CurrentDocument.DeleteQuery "Query1"
```

1-2-3: AddSeparator method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS;',0)} See list of classes

{button ,AL(`H_123_ADDSEPARATOR_METHOD_EXSCRIPT',1)} See example

Adds a separator to a menu.

Syntax

object.**AddSeparator** *position*

Parameters

position

Long. The separator's position in the menu.

Value	Description
Positive integer	The new separator's position in the menu, counting forward from the beginning. The value 1 means after the first menu item. If *position* exceeds the last existing item's position, the new separator is placed at the end.
Zero	Place the new separator before the first menu item.
Negative integer	The new separator's position in the menu, counting backward from the end. The value –2 means before the last menu item.

Return values

None

Usage

Use a separator to categorize groups of menu items.

If the separator is in the last position, it is not visible.

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_CHECKITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_REPLACEMENU_METHOD_MEMDEF;H_123_RESETMENUBAR_METHOD_MEMDEF;H_123_UNCHECKITEM_METHOD_MEMDEF;H_123_GETMENU_METHOD_MEMDEF;H_123_GETMENUPOSITION_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_CURRENTMENUBAR_PROPERTY_MEMDEF;',0)} See related topics

```
' Example: AddSeparator method, AddItem method, and NewMenu method
' Create a menu, add 3 menu items, and add a separator after the first.
' For more context on this example, see the example for the AddItem method.
' You need item scripts "itemHandler1" through "itemHandler3" available to use this
code.
' For an example of such a handler, see the example for the CheckItem method.
Dim menu1 As Menu
Set menu1 = CurrentApplication.NewMenu
menu1.AddItem 1, "Menu pick 1", "Prompt 1", ThisDocument, "itemHandler1"
menu1.AddItem 2, "Menu pick 2", "Prompt 2", ThisDocument, "itemHandler2"
menu1.AddItem 3, "Menu pick 3", "Prompt 3", ThisDocument, "itemHandler3"
menu1.AddSeparator 1
```

1-2-3: AddToSelection method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_PLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDTOSELECTION_METHOD_EXSCRIPT;H_123_SELECTION_PROPERTY_EXSCRIPT;',1)} See example

Adds an object to the selection for the current file, without removing currently selected objects.

Syntax

object.**AddToSelection**

Parameters

None

Return values

None

{button ,AL(`H_123_CLEARSELECTION_METHOD_MEMDEF;H_123_COPYSELECTION_METHOD_MEMDEF;H_123_CUTSELECTION_METHOD_MEMDEF;H_123_EXTENDSELECTFROMTAB_METHOD_MEMDEF;H_123_EXTENDSELECT_METHOD_MEMDEF;H_123_EXTENDWORKSHEETSELECTIONBACK_METHOD_MEMDEF;H_123_EXTENDWORKSHEETSELECTIONFORWARD_METHOD_MEMDEF;H_123_GETSELECTION_METHOD_MEMDEF;H_123_REMOVEFROMSELECTION_METHOD_MEMDEF;H_123_SELECT_METHOD_MEMDEF;H_123_SELECTALL_METHOD_MEMDEF;H_123_SELECTALLSHEETS_METHOD_MEMDEF;H_123_ACTIVE_PROPERTY_MEMDEF;H_123_ISSELECTABLE_PROPERTY_MEMDEF;H_123_ISSELECTED_PROPERTY_MEMDEF;H_123_SELECTION_PROPERTY_MEMDEF;H_123_SELECTED_EVENT_MEMDEF',0)} See related topics

```
' Example: AddToSelection, Print, and RemoveFromSelection methods
' Create two ranges and select them. Then print them and unselect the second range.
Dim range1 As Range, range2 As Range
Set range1 = Bind("A:A1..A:A10")

Set range2 = Bind("A:B10..A:B20")
' Select A:A1..A:A10, then add A:B10..A:B20 to the selection.

range1.Select
range2.AddToSelection
' Print both ranges in list.

Print(Selection.Name)
' Remove range 2 from the selection.
range2.RemoveFromSelection
```

1-2-3: AddVersion method

{button ,AL(`H_123_VERSIONGROUP_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDVERSION_METHOD_EXSCRIPT',1)} See example

Adds a version to this version group.

Syntax

versiongroup.**AddVersion** *rangename, versionname,* [*lastmodifier*]

Parameters

rangename

String. The name of the range that contains the version that you want to add to the version group.

versionname

String. The name of the version that you want to add to the version group.

lastmodifier

(Optional) String. The name of the last person to modify this version.

Return values

None

Usage

You can put only one version of each named range into a single version group.

{button ,AL(`H_123_DELETEVERSION_METHOD_MEMDEF;H_123_MERGEVERSIONS_METHOD_MEMDEF;H_123_NEWVERSIONGROUP_METHOD_MEMDEF;H_123_NEWVERSION_METHOD_MEMDEF;H_123_REMOVEVERSION_METHOD_MEMDEF;H_123_REPORTVERSION_METHOD_MEMDEF;H_123_VERSIONGROUP_METHOD_MEMDEF;H_123_VERSIONS_METHOD_MEMDEF;H_123_VERSION_METHOD_MEMDEF;H_123_CURRENTVERSION_PROPERTY_MEMDEF;H_123_LASTVERSIONGROUP_PROPERTY_MEMDEF;H_123_SHOWVERSIONBORDERS_PROPERTY_MEMDEF;H_123_VERSIONBORDERSVISIBLE_PROPERTY_MEMDEF;H_123_VERSIONBORDERVISIBLE_PROPERTY_MEMDEF',0)} See related topics

```
' Example: Background, BackColor, Colors properties; AddVersion,
'          CreateRangeName, DeleteVersion, DeleteVersionGroup, NewVersion,
'          NewVersionGroup, RemoveVersion methods
' This example creates 2 range names and a version for each.
' It then puts the versions in a version group.
' First, create the ranges.
CurrentDocument.CreateRangeName "North Sales", [A:B2..A:B10]
CurrentDocument.CreateRangeName "South Sales", [A:C2..A:C10]
' Next, make a second version for each range.
[North Sales].NewVersion("Version 2")
[South Sales].NewVersion("Version 2")
' Do something to make Version 2 different. For example, set the background color.
Set [North Sales].Background.BackColor = CurrentApplication.Colors("light yellow")
Set [South Sales].Background.BackColor = CurrentApplication.Colors("pale blue")
' Create a version group and add the versions to it.
CurrentDocument.NewVersionGroup("Version Group A")
[Version Group A].AddVersion "North Sales", "Version 2"
[Version Group A].AddVersion "South Sales", "Version 2"
' After you're finished with the version, you can remove it or delete it.
' If necessary, remove a version from the version group.
[Version Group A].RemoveVersion "South Sales"
' If necessary, delete the version.
[South Sales.Version 2].DeleteVersion
' When finished, you can also delete Version Group A from the document.
[Version Group A].DeleteVersionGroup
```

1-2-3: AppendRecords method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_APPENDRECORDS_METHOD_EXSCRIPT',1)} See example

Appends records from the specified source range to this database table range. The range that is the object of this method must be a database table in the worksheet or an external database table.

Syntax

range.**AppendRecords** *sourcerange*

Parameters

sourcerange

Variant. The range of records to append to this range. The range to append must be organized as a database table.

Return values

None

{button ,AL(`H_123_DELETERECORDS_METHOD_MEMDEF;H_123_FINDRECORDS_METHOD_MEMDEF;H_123_CREATETABLE_METHOD_MEMDEF',0)} See related topics

```
' Example: AppendRecords and CreateRangeName methods
' Create 2 database tables and then append the second to the first.
' Create a new file
Dim doc1 As Document
Set doc1 = CurrentApplication.NewDocument("Doc_1")
' Create the first database table.
CurrentDocument.CreateRangeName "EmpTable1", [A:B2..A:C10]
[A:B2].Contents = "Name"
[A:C2].Contents = "Employee ID"
[A:B3].Contents = "Smith, John"
[A:C3].Contents = "1234"
' Create the second database table.
.NewSheet $After,1,True
CurrentDocument.CreateRangeName "EmpTable2", [B:B2..B:C10]
[B:B2].Contents = "Name"
[B:C2].Contents = "Employee ID"
[B:B3].Contents = "Doe, Jane"
[B:C3].Contents = "5678"
' Append the second table to the first table.
[EmpTable1].AppendRecords([EmpTable2])
[EmpTable1].Goto
```

1-2-3: ArrangeIcons method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;',0)} See list of classes

Arranges the icons within the application window.

Syntax

applicationwindow.**ArrangeIcons**

Parameters

None

Return values

None

1-2-3: AutoSmartSum method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_AUTOSMARTSUM_METHOD_EXSCRIPT',1)} See example

Generates a sum cell for a range of numbers near a cell containing the string "Total".

Syntax

*range.***AutoSmartSum**

Parameters

None

Return values

None

Usage

This method is activated when you type in a cell the word Total or Totals. These labels are case-insensitive. From the location of the cell with "Total" in it, this method searches for nearby data that should be summed, up to 10 rows or columns away. If it finds a range of data that seems suitable, it puts @Sum for that range in the appropriate cells.

This command is not generated if you just put the string "Total" in a cell via script.

AutoSmartSum is enabled when the Application.UsingTotalToAutoSum property is set to True, which is the default setting.

{button ,AL(`H_123_SMARTSUM_METHOD_MEMDEF;H_123_USINGTOTALTOAUTOSUM_PROPERTY_MEMDEF',0)} See related topics

```
' Example: AutoSmartSum method
' Run AutoSmartSum on a column of 3 cells.
' Enter values to be summed in B1..B3.
[A:B1].Contents = "1"
[A:B2].Contents = "2"
[A:B3].Contents = "3"
' Enter the "Total" string for AutoSmartSum in A4.
[A:A4].Contents = "Total"
' AutoSmartSum on cell A4 containing "Total" puts @SUM(B1..B3) in B4.
[A:A4].AutoSmartSum
' Output: The value "6" will appear in cell B4.
```

1-2-3: Backsolve method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_BACKSOLVE_METHOD_EXSCRIPT',1)} See example

Makes a formula cell equal to a specified value, by changing the values of specified adjustable cells that are used by the formula.

Syntax

document.**Backsolve** *makecell, cellvalue, adjustablecells*

Parameters

makecell

Variant. The cell containing the formula.

cellvalue

Variant. The value to which the *makecell* formula output is to be changed by adjusting the values in the *adjustablecells* range.

adjustablecells

Variant. The range of cells you want to change, to cause the formula to result in *cellvalue*.

Return values

None

```
' Example: BackSolve method

' Modify the value in cell A1

' so that the formula in cell B1 returns the value 20.

[B1].Contents = "+A1*5"
CurrentDocument.BackSolve [B1],20,[A1]

' Results:

' A1 will contain 4.

' The formula in B1 will show 20.
```

1-2-3: Bounds method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_PLOT_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_BOUNDS_METHOD_EXSCRIPT',1)} See example

Redraws the graphic object using (*left*, *top*) as the top left corner and (*right*, *bottom*) as the bottom right corner. All units are one-twentieth of a point (twips).

Syntax

document.*object*.**Bounds** *left, top, right, bottom*

Parameters

left

Long. Left position of the object's bounding rectangle, in units of twips.

top

Long. Top position of the object's bounding rectangle, in units of twips.

right

Long. Right position of the object's bounding rectangle, in units of twips.

bottom

Long. Bottom position of the object's bounding rectangle, in units of twips.

Return value

None

{button ,AL(`H_123_MOVE_METHOD_MEMDEF;H_123_RESIZE_METHOD_MEMDEF',0)} See related topics

```
' Example: Bounds and NewEllipse methods
' Create an ellipse in a 1"x2" box and then redraw it in a 0.5"x1" box.
' First, draw an ellipse on sheet A.
Dim myEllipse As Ellipse
Set myEllipse = [A].NewEllipse(1440, 1440, 4320, 2880)
' Redraw the ellipse in a smaller bounding rectangle.
myEllipse.Bounds 1440, 1440, 2880, 2160
```

1-2-3: Calc method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_CALC_METHOD_EXSCRIPT',1)} See example

Recalculates the formulas in open files.

Syntax

object.**Calc**

Parameters

None

Return values

None

{button ,AL(`H_123_RANGERECALC_METHOD_MEMDEF;H_123_CALCITERATIONS_PROPERTY_MEMDEF;H_123_CALCMODE_PROPERTY_MEMDEF;H_123_CALCORDER_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CalcMode property; Calc method
' Set up a summed range, change one its cells, and recalculate the sum.
' First, create a summed range.
[A:B2].Contents = "1"

[A:B3].Contents = "2"

[A:B4].Contents = "3"

[A:B5].Contents = "@Sum(B2..B4)"
' Set Calc mode to manual,
' to can control when recalculation occurs.
CurrentDocument.CalcMode = $Manual
' B5 now shows 6.  Now change B2.
[A:B2].Contents = "2"
' To see the effect of this, you must recalculate,
' because .CalcMode was set to $Manual.
CurrentApplication.Calc
' B5 now shows 7.
```

1-2-3: Cascade method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_CASCADE_METHOD_EXSCRIPT',1)} See example

Cascades the workbook windows when more than one window is open.

Syntax

applicationwindow.**Cascade**

Parameters

None

Return values

None

{button ,AL(`H_123_TILE_METHOD_MEMDEF;H_123_TILEHORIZONTAL_METHOD_MEMDEF;H_123_TILEVERTICAL_METHOD_MEMDEF',0)} See related topics

```
' Example: ApplicationWindow property; Cascade method
' Create two workbook windows and cascade them.
Dim appWindow As ApplicationWindow
Dim doc1 As Document
Dim doc2 As Document
Set appWindow = CurrentApplication.ApplicationWindow
Set doc1 = CurrentApplication.NewDocument("Document #1")
Set doc2 = CurrentApplication.NewDocument("Document #2")
' Cascade the workbook windows.
appWindow.Cascade
```

1-2-3: Cell method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_CELL_METHOD_EXSCRIPT',1)} See example

Returns the single-cell range for a specified row, column, and sheet. You specify the row, column, and sheet using indexes that are zero-based from the top leftmost cell of the first sheet of the range on which you run this method. The top leftmost cell of the first sheet thus has the indexes (0, 0, 0).

Syntax

Set *cellrange* **=** *range***.Cell**(*row, column,* [*sheet*])

Parameters

row

Long. The zero-based row of the cell.

column

Long. The zero-based column of the cell.

sheet

(Optional) Long. The zero-based sheet of the cell.

Return values

Range. The desired cell range.

Usage

The specified indexes must designate a cell within the range on which you run this method.

{button ,AL(`H_123_CELLS_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CoordinateString property; Cell method
' Get a single-cell range from its row and column titles in row 1 and column A.

' Search titles.
Dim rowTitle As String       ' The cell row title to find
Dim colTitle As String       ' The cell column title to find

' Result variables.
Dim cellFound As Variant     ' The cell found by this routine
Dim rowFound As Variant      ' True if a row with rowTitle in column A was found
Dim colFound As Variant      ' True if a column with colTitle in row 1 was found
rowFound = False
colFound = False

' Set up test data.
rowTitle = "markRow"
colTitle = "markCol"
[A3].Contents = rowTitle
[D1].Contents = colTitle

' Find the row title cell that matches rowTitle.
Dim rowRange As Range        ' The range containing all the row titles (column A)
Dim row As Long              ' Index of the row being searched
Dim rowCell As Variant       ' The single cell being searched
Set rowRange = Bind("A1..A8192")
row = 0                      ' Index is zero-based
Do While row < 8192
   Set rowCell = rowRange.Cell(row, 0, 0)
   ' Remove the label-prefix character from the cell contents
   ' and compare to the desired row title.
   If Mid(rowCell.Contents, 2) = rowTitle Then
      rowFound = True
      Exit Do
      ' You can add code here to limit the number of blank cells examined,
      ' when rowTitle is not found ...
   End If
   row = row + 1
Loop

' Find the column title cell that matches colTitle
Dim colRange As Range        ' The range containing all the column titles (row 1)
Dim col As Long              ' Index of the column being searched
Dim colCell As Variant       ' The single cell being searched
Set colRange = Bind("A1..IV1")
col = 0                      ' Index is zero-based
Do While col < 256
   Set colCell = colRange.Cell(0, col, 0)
   ' Remove the label-prefix character from the cell contents
   ' and compare to the desired column title.
   If Mid(colCell.Contents, 2) = colTitle Then
      colFound = True
      Exit Do
      ' You can add code here to limit the number of blank cells examined,
      ' when colTitle is not found ...
   End If
   col = col + 1
Loop
```

```
' Get the single-cell matching range that was found.
If rowFound = True And colFound = True Then
   Set cellFound = [A1..IV8192].Cell(row, col, 0)
Else
   Set cellFound = Null
End If
Messagebox "The specified cell is " & cellFound.CoordinateString, MB_OK +
MB_ICONINFORMATION
' Test result is cellFound = [D3].
```

1-2-3: CheckItem method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS;',0)} See list of classes

{button ,AL(`H_123_CHECKITEM_METHOD_EXSCRIPT',1)} See example

Puts a check mark before the name of a specified item on the menu.

Syntax

object.**CheckItem** *position*

Parameters

position

Long. The menu position of the item to check.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position.

Return values

None

Usage

Use a check mark to indicate that the corresponding menu item is set, true, or on. Typically, you call this method in the Click event for the menu item. This method only works on items created using LotusScript.

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_ADDSEPARATOR_METHOD_MEMDEF;H_123_DISABLEITEM_METHOD_MEMDEF;H_123_ENABLEITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_UNCHECKITEM_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CheckItem method, UncheckItem method
' Handler for a menu item that sets a property True or False
' and checks the menu item when the property is set True.
' See the example for the AddMenu method for an example of
' how to hook this item handler up to a displayed menu.

' In the Globals section, Dim the variables.
Dim settingValue As Variant         ' The Boolean property to be set.
Dim settingPos As Integer           ' The menu position of the property item.
Dim menu As Menu                    ' The menu containing the item.

Sub MenuItemInitialize              ' For testing only.
   settingValue = False
   settingPos = 1
   Set menu = CurrentApplication.NewMenu
End Sub

Sub mySettingHandler
   ' Set the property.
   settingValue = Not settingValue
   ' Set the menu item check mark.
   If settingValue = True  Then menu.CheckItem   settingPos
   If settingValue = False Then menu.UncheckItem settingPos
End Sub
```

1-2-3: ClearOutline method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_CLEAROUTLINE_METHOD_EXSCRIPT',1)} See example

Clears the entire outline for the sheet. Clearing the outline fully promotes and expands all outlined rows and columns in the sheet and turns off the display of the outline frames for the rows and columns.

Syntax

sheet.**ClearOutline**

Parameters

None

Return values

None

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_COLLAPSEALL_METHOD_MEMDEF;H_123_EXPANDALL_METHOD_MEMDEF',0)} See related topics

```
' Example: ClearOutline method, DemoteRow method
' Demote some rows and then clear the outline in sheet A.
' First, demote rows 2 through 10 in Sheet A.
[A:A2..A:A10].DemoteRow
' Reverse the demotion by clearing the outline.
[A].ClearOutline
```

1-2-3: ClearRangeNames method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_CLEARRANGENAMES_METHOD_EXSCRIPT',1)} See example

Deletes all range names in a file.

Syntax

document.**ClearRangeNames**

Parameters

None

Return values

None

{button ,AL(`H_123_CREATERANGENAME_METHOD_MEMDEF;H_123_DELETERANGENAME_METHOD_MEMDEF;H_123_CREATERANGENAMEFROMLABEL_METHOD_MEMDEF;H_123_CREATERANGENAMETABLE_METHOD_MEMDEF;H_123_ISRANGENAMED_METHOD_MEMDEF;H_123_NAME_PROPERTY_MEMDEF',0)} See related topics

```
' Example: ClearRangeNames, CreateRangeName methods
' Create, use, and clear some range names.
' First, create the named ranges.
CurrentDocument.CreateRangeName "Range 1", [A:B2..A:B10]
CurrentDocument.CreateRangeName "Range 2", [A:C2..A:C10]
' Do something with these ranges using their range names ...
' Then delete them all.
CurrentDocument.ClearRangeNames
```

1-2-3: ClearSplits method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_NEXTSPLIT_METHOD_EXSCRIPT',1)} See example

Clears existing splits in the workbook window.

Syntax

document.**ClearSplits**

Parameters

None

Return values

None

{button ,AL(`H_123_VIEWSPLITHEIGHT_PROPERTY_MEMDEF;H_123_VIEWSPLITSTYLE_PROPERTY_MEMDEF;H_123_VIEWSPLITWIDTH_PROPERTY_MEMDEF;H_123_NEXTSPLIT_METHOD_MEMDEF',0)} See related topics

1-2-3: Clear method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DOCUMENT_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_CLEAR_METHOD_EXSCRIPT;H_123_SHOWVERSIONBORDERS_PROPERTY_EXSCRIPT;',1)} See example

Deletes the object without placing it on the Clipboard. For a Range object, clears the specified data from the range.

Syntax

object.**Clear** [*clearchoice*]

Parameters

clearchoice

(Optional) Variant (RangeClearChoice enumeration). Valid only for Range objects. Specifies what data to clear from the range. The following table lists the allowed values for this parameter. These values can be combined. The default is to clear the cell contents of the range.

Value	Description
$ClearData	Clear the cell contents
$ClearStyle	Clear the style
$ClearFormat	Clear the data format
$ClearBorder	Clear the cell border
$ClearScripts	Clear the scripts
$ClearComment	Clear the cell comment

Return values

None

{button ,AL(`H_123_CLEAROUTLINE_METHOD_MEMDEF;H_123_CLEARRANGENAMES_METHOD_MEMDEF;H_123_CLEARSPLITS_METHOD_MEMDEF;H_123_CLEARTRANSCRIPT_METHOD_MEMDEF;',0)} See related topics

```
' Example: Background, Color, ColorName and Pattern properties;
'          Clear and NewRectangle methods

' Enter data into cell B2 and then clear it.
[A:B2].Contents = "100"
Msgbox "Clear the data."
[A:B2].Clear $ClearData

' Create a rectangle and then clear it from the sheet.
Dim rect1 As Rectangle
Set rect1 = [A].NewRectangle(100, 100, 1000, 1000)
' Do something with the rectangle. For example, set its background.
[Rectangle 1].Background.Color.ColorName = "neon green"
[Rectangle 1].Background.Pattern = $BowlingBalls
' When finished, delete the rectangle.
Msgbox "Clear the rectangle."
rect1.Clear
```

1-2-3: CloseAll method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_CLOSEALL_METHOD_EXSCRIPT',1)} See example

In a 1-2-3 add-in, closes all open documents, saving changes if you so specify. This method is only for use in scripts that will be compiled into a 1-2-3 add-in file.

Syntax

application.**CloseAll** *savechanges*

Parameters

savechanges

Variant (Boolean). Indicates whether to save any changes in the documents (value True) or not to save them (value False).

Return values

None

{button ,AL(`H_123_NEWDOCUMENT_METHOD_MEMDEF;H_123_OPEN_METHOD_MEMDEF;H_123_OPENDOCUMENT_METHOD_MEMDEF;H_123_CLOSE_METHOD_MEMDEF',0)} See related topics

```
' Example: CloseAll method
' Save all changes in the currently open files and close them.

' First, create a document and modify it.
Dim scratchfile As Document
Set scratchfile = CurrentApplication.NewDocument
[A:B2].Contents = "testdata"
' Save all changes in the document and close it.
CurrentApplication.CloseAll True
' This brings up the SaveAs dialog box to prompt the user
' for the filenames of any new files that have never been saved.
' After the user supplies the filenames, 1-2-3 closes all open files.
```

1-2-3: Close method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_CLOSE_METHOD_EXSCRIPT',1)} See example

Closes the window, document, or application.

Syntax

object.**Close** [*savechanges*]

Parameters

savechanges

(Optional) Variant. A Boolean value that indicates whether to save any changes that have been made to the file. If you don't pass in this argument and changes have been made in a file with a single open document window, the Save Changes dialog box appears. The dialog box doesn't appear when you run this method on an Application object that has no open files, or on a DocWindow object that is not the only one open for a file.

Return values

None

Usage

When you close an application window and the application is serving objects to a client, the application remains active (but not visible) until all served objects are released. At that point, the application ends.

{button ,AL(`H_123_NEWDOCUMENT_METHOD_MEMDEF;H_123_OPEN_METHOD_MEMDEF;H_123_OPENDOCUMENT_METHOD_MEMDEF;H_123_CLOSEALL_METHOD_MEMDEF',0)} See related topics

```
' Example: Close method
' Example 1: Close the current document and save any changes.
CurrentDocument.Close True
' Example 2: Close document window 3 in the document <<Untitled>>.
[<<Untitled>>.Window 3].Close
```

1-2-3: CollapseColumn method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_COLLAPSECOLUMN_METHOD_EXSCRIPT',1)} See example

Collapses all expanded columns whose parents (summary columns) are within this range.

Syntax

range.**CollapseColumn**

Parameters

None

Return values

None

Usage

This method doesn't work on 3D ranges.

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_COLLAPSEALL_METHOD_MEMDEF;H_123_EXPANDALL_METHOD_MEMDEF',0)} See related topics

```
' Example: CollapseColumn method, DemoteColumn method, ExpandColumn method
' Demote column D, collapse column E above it,
' and then expand column E.
[A:D1].DemoteColumn 1
[A:E1].CollapseColumn
' Column D is no longer visible.
' To see column D again, expand column E.
[A:E1].ExpandColumn
```

1-2-3: CollapseRow method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_COLLAPSEROW_METHOD_EXSCRIPT',1)} See example

Collapses all expanded rows whose parents (summary rows) are in the range.

Syntax

range.**CollapseRow**

Parameters

None

Return values

None

Usage

This method doesn't work on 3D ranges.

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_COLLAPSEALL_METHOD_MEMDEF;H_123_EXPANDALL_METHOD_MEMDEF',0)} See related topics

```
' Example: CollapseRow method, DemoteRow method, ExpandRow method
' Demote row 4, collapse the row 5 below it,
' and then expand the row.
[A:A4].DemoteRow 1
[A:A5].CollapseRow
' Row 4 is no longer visible.
' To see row 4 again, expand row 5.
[A:A5].ExpandRow
```

1-2-3: ColorFromRGB method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_COLORFROMRGB_METHOD_EXSCRIPT',1)} See example

Creates a Color object whose RGB value is closest to the specified RGB value, from the colors available in the application.

Syntax

color **=** *application***.ColorFromRGB**(*rgbvalue*)

Parameters

rgbvalue

Long. The RGB value to be converted to a Color object. The format for this value is &H00RRGGBB&, where RR, GG, and BB are the red, green, and blue intensities, each varying from 00 to FF.

Return values

A Color object.

{button ,AL(`H_123_COLORS_PROPERTY_MEMDEF',0)} See related topics

```
' Example: Background and BackColor properties; ColorFromRGB method
' Create a primary blue Color object from a primary blue RGB value.
Dim myColor As Color
Dim rgbValue As Long
rgbValue = &H000000FF&
Set myColor = CurrentApplication.ColorFromRGB(rgbValue)
' Apply the color to a range.
Set [A:A2..A:A10].Background.BackColor = myColor
```

1-2-3: Connect method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Connects to an external database table and creates an external range that can be used in a query.

Syntax

document.**Connect** *drivername,* [*driveruserid*], [*driverpassword*], [*connectstring*], *databasename,* [*databaseuserid*], [*databasepassword*], [*ownername*], *tablename,* [*rangename*]

Parameters

drivername

String. The name of the driver to use for the connection (for example, "DBase_IV").

driveruserid

(Optional) String. The database driver user ID.

driverpassword

(Optional) String. The database driver password.

connectstring

(Optional) String. A connection string to pass to the driver. Use this to specify additional information that may be needed to connect to the database.

databasename

String. The name and path of the external database to which you want to connect (for example, "\lotus\work\123\database\staff").

databaseuserid

(Optional) String. The database user ID.

databasepassword

(Optional) String. The database password.

ownername

(Optional) String. The name of the owner of the table.

tablename

String. The name of the table in the external database (for example, "employee.dbf").

rangename

(Optional) String. The name of the external range to create. If you don't supply this, the table name is used.

Return values

None

{button ,AL(`H_123_DISCONNECT_METHOD_MEMDEF',0)} See related topics

1-2-3: CopySelection method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_COPYSELECTION_METHOD_EXSCRIPT',1)} See example

Copies the selected objects to the Clipboard.

Syntax

document.**CopySelection**

Parameters

None

Return values

None

{button ,AL(`H_123_CUTSELECTION_METHOD_MEMDEF;H_123_GETSELECTION_METHOD_MEMDEF;H_123_REMOVEFROMSELECTION_METHOD_MEMDEF;H_123_SELECT_METHOD_MEMDEF;H_123_SELECTALL_METHOD_MEMDEF;H_123_SELECTALLSHEETS_METHOD_MEMDEF;H_123_SELECTION_PROPERTY_MEMDEF;',0)} See related topics

```
' Example: CopySelection, Select, and Paste methods
' Copy and paste a selection.

' First, select a range.
[A:B2..A:B10].Select
' Copy it to the Clipboard.
CurrentDocument.CopySelection
' Paste it somewhere else.
[A:D2..A:D10].Paste
```

1-2-3: CopySQLToClipboard method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_COPYSQLTOCLIPBOARD_METHOD_EXSCRIPT',1)} See example

Copies the SQL statement that represents the current query to the Clipboard.

Syntax

query.**CopySQLToClipboard**

Parameters

None

Return values

None

{button ,AL(`H_123_NEWQUERY_METHOD_MEMDEF;H_123_QUERYSORTDEFINEKEY_METHOD_MEMDEF',0)} See related topics

```
' Example: CopySQLToClipboard, OpenDocument, and Paste methods
Sub CopySQLToCLIP
   ' Open a workbook containing a table.
   CurrentApplication.OpenDocument "e:\data\123\employee.123"

   ' Set up a query, copy it to the Clipboard, and paste it.
   CurrentDocument.NewQuery "Query1", "A:A1..A:F11"
   ' Run the query.
   [Query1].OutputLocation = "A:A15"
   [Query1].Refresh

   ' Enter your SQL query into Query #1 ...
   Msgbox "Enter your SQL query into the query."
   [Query1].RemoveSelectField "LAST"
   [Query1].AddSelectField "LAST", "DOH"

   ' Copy the query to the Clipboard.
   [Query1].CopySQLToClipboard

   ' Paste the SQL string from the Clipboard into A:A13.
   [A:A13].Paste

   ' Output =
      ' SELECT EMPID, LAST, FIRST, DOH, SALARIED, DEPTNUM FROM A:A1..A:F11
End Sub
```

1-2-3: CopyToClipboard method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DOCUMENT_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_COPYTOCLIPBOARD_METHOD_EXSCRIPT',1)} See example

Copies the object to the Clipboard.

Syntax

object.**CopyToClipboard** [*format*]

Parameters

format

(Optional) Variant (ClipboardFormat enumeration). The format in which to copy to the Clipboard. The following table lists the allowed values for this parameter.

Value	Description
$NativeFormat	Native format
$RichTextFormat	Rich text format
$BitmapFormat	Bitmap format
$PictureFormat	Picture format
$LotusChartFormat	Lotus Chart format
$EmbedSourceFormat	Embed source format
$EmbeddedObjectFormat	Embedded object format
$DIBFormat	Device-independent bitmap
$TextFormat	Text
$WK1Format	1-2-3 for DOS, Release 2
$WK3Format	1-2-3 for Windows Release 1; 1-2-3 for DOS Releases 3, 4

Return values

None

{button ,AL(`H_123_PASTE_METHOD_MEMDEF',0)} See related topics

```
' Example: CopyToClipboard, Paste methods
' Copy a range's data to the Clipboard and paste it into another range.
[A:B2].Contents = "123"
[A:B2..A:B10].CopyToClipboard
[A:C2..A:C10].Paste
```

1-2-3: CreateComputedField method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_CREATECOMPUTEDFIELD_METHOD_EXSCRIPT',1)} See example

Adds a field containing a specified formula to the query.

Syntax

dataquery.**CreateComputedField** *formula,* [*alias*]

Parameters

formula

String. The formula in 1-2-3 syntax. For more information, search on "Formulas, guidelines" in the Help Index.

alias

(Optional) String. A field alias for the computed field. An alias replaces the formula in the computed field name cell.

Return values

None

{button ,AL(`H_123_DELETECOMPUTEDFIELD_METHOD_MEMDEF;H_123_ADDSELECTFIELD_METHOD_MEMDEF;H_GUIDELINES_FOR_ENTERING_FORMULAS_OVER@SS1N60EN.HLP',0)} See related topics

```
' Example: OutputLocation property; CreateComputedField, DeleteComputedField,
'          DeleteQuery, NewQuery, and Refresh methods;
' Set up a new query, refresh it, and then delete it.
'
' Set up the query.
[A:B2].Contents = "Item"
[A:C2].Contents = "Price"
[A:D2].Contents = "Shipping"
[A:B3].Contents = "1"
[B3].DragAndFill [B3..D5]
CurrentDocument.NewQuery "Query 1", "A:B2..A:D5"
' Run the query.
[Query 1].OutputLocation = "A:B8"
[Query 1].Refresh
' Do something with the query. For example, create a computed field.
[Query 1].CreateComputedField "(Price)+(Shipping)", "Total Price"
' If necessary, delete the computed field.
If 6 = Messagebox("Delete the computed field?", 4 + 32, "Computed field example") Then
    [Query 1].DeleteComputedField "(Price)+(Shipping)"
End If
' If necessary, delete the query.
If 6 = Messagebox("Delete the query?", 4 + 32, "Computed field example") Then
    CurrentDocument.DeleteQuery "Query 1"
End If
```

1-2-3: CreateRangeNameFromLabel method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CREATERANGENAMEFROMLABEL_METHOD_EXSCRIPT',1)} See example

Assigns an existing range of labels as the range names for a group of single-cell ranges immediately above, below, to the right of, or to the left of the label range.

Syntax

document.**CreateRangeNameFromLabel** *range, rangelabeldirection*

Parameters

range

Variant. The range that contains the labels you want to assign as range names for adjacent cells.

rangelabeldirection

Variant (RangeNameLabelDir enumeration). Indicates which single-cell ranges to name using adjacent labels. The following table lists the allowed values for this parameter.

Value	Description
$LabelLeft	Names cells to the left of the labels
$LabelRight	Names cells to the right of the labels
$LabelUp	Names cells above the labels
$LabelDown	Names cells below the labels

Return values

None

{button ,AL(`H_123_CLEARRANGENAMES_METHOD_MEMDEF;H_123_CREATERANGENAME_METHOD_MEMDEF;H_123_DELETERANGENAME_METHOD_MEMDEF;H_123_CREATERANGENAMETABLE_METHOD_MEMDEF;H_123_ISRANGENAMED_METHOD_MEMDEF;H_123_NAME_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CreateRangeNameFromLabel and DragAndFill methods
' Create a column of labels and convert them to range names.
' First, create a range label in cell B2.
[A:B2].Contents = "Range_1"
' Fill the rest of the column B cells through row 10 with sequential names.
[A:B2].DragAndFill [A:B2..A:B10]
' Create range names from the labels.
CurrentDocument.CreateRangeNameFromLabel [A:B2..A:B10], $LabelRight
```

1-2-3: CreateRangeNameTable method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CREATERANGENAMETABLE_METHOD_EXSCRIPT',1)} See example

Creates a two-column table with the names of all defined ranges in the current file listed alphabetically in the left column, and the corresponding range addresses listed in the right column.

Syntax

document.**CreateRangeNameTable** *tablelocation*

Parameters

tablelocation

Variant. The name or address of the range where you want to create the table of range names and addresses. Specify either the entire range or only the first cell.

Return values

None

Usage

The table occupies two columns and as many rows as there are range names, plus one blank row. 1-2-3 writes over any existing data in the table range.

{button ,AL(`H_123_CLEARRANGENAMES_METHOD_MEMDEF;H_123_CREATERANGENAME_METHOD_MEMDEF;H_123_CREATERANGENAMEFROMLABEL_METHOD_MEMDEF;H_123_ISRANGENAMED_METHOD_MEMDEF;H_123_NAME_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CreateRangeName and CreateRangeNameTable methods
' Create a range name table starting on sheet A, in cell F2.
' First, create two range names.
CurrentDocument.CreateRangeName "Range_1", [A:A2..A:A5]
CurrentDocument.CreateRangeName "Range_2", [A:B2..A:B5]
' Place the range names in a table starting in cell A:F2.
CurrentDocument.CreateRangeNameTable [A:F2]
' Output: Two range names and addresses at starting in cell A:F2.
```

1-2-3: CreateRangeName method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETERANGENAME_METHOD_EXSCRIPT;H_123_SHOWVERSIONBORDERS_PROPERTY_EXSCRIPT;',1)} See example

Assigns a name to a range.

Syntax

document.**CreateRangeName** *rangename, range*

Parameters

rangename

String. The name you want to assign to the range.

range

Range. The range that you want to name.

Usage

Range names must be no more then 15 characters and must start with a letter (A–Z). For other name restrictions, search on "Naming conventions" in the 1-2-3 Help Index.

{button ,AL(`H_123_CLEARRANGENAMES_METHOD_MEMDEF;H_123_CREATERANGENAMEFROMLABEL_METHOD_MEMDEF;H_123_CREATERANGENAMETABLE_METHOD_MEMDEF;H_123_ISRANGENAMED_METHOD_MEMDEF;H_123_NAME_PROPERTY_MEMDEF',0)} See related topics

1-2-3: CreateTable method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Sets up the structure for and connects to a new table in an external database.

Syntax

document.**CreateTable** *drivername,* [*driveruserid*], [*driverpassword*], [*connectstring*], *databasename,* [*databaseuserid*], [*databasepassword*], [*ownername*], *tablename,* [*rangename*], [*createstring*], *modelrange*

Parameters

drivername

String. The name of the driver to use for the connection to the database (for example, "dBASE_IV").

driveruserid

(Optional) String. The database driver user ID.

driverpassword

(Optional) String. The database driver password

connectstring

(Optional) String. A connection string to pass to the driver. Use this to specify additional information that may be needed to connect to the database.

databasename

String. The name and path of the database to which you want to connect (for example, "\lotus\work\database\staff").

databaseuserid

(Optional) String. The database user ID.

databasepassword

(Optional) String. The database password.

ownername

(Optional) String. The name of the owner of the table.

tablename

String. The name of the table in the external database (for example, "employee.dbf").

rangename

(Optional) String. The name of the external range to create for the table. If you don't supply this, the table name is used.

createstring

(Optional) String. The table creation string to be passed to the driver.

modelrange

Range. The database table range that contains the data from which the new table is to be created.

Return values

None

{button ,AL(`H_123_BASESOURCETABLE_PROPERTY_MEMDEF;H_123_APPENDRECORDS_METHOD_MEMDEF;H_123_DELETERECORDS_METHOD_MEMDEF;H_123_SETMAXRECORDS_METHOD_MEMDEF;H_123_NEWQUERYTABLE_METHOD_MEMDEF;H_123_WHATIFTABLE1_METHOD_MEMDEF;H_123_WHATIFTABLE2_METHOD_MEMDEF;H_123_WHATIFTABLE3_METHOD_MEMDEF;H_123_SOURCETABLES_PROPERTY_MEMDEF',0)} See related topics

1-2-3: CutSelection method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CUTSELECTION_METHOD_EXSCRIPT',1)} See example

Cuts the selected objects to the Clipboard.

Syntax

document.**CutSelection**

Parameters

None

Return values

None

{button ,AL(`H_123_CUT_METHOD_MEMDEF;H_123_PASTE_METHOD_MEMDEF',0)} See related topics

```
' Example: CutSelection, Paste methods
' Select, cut, and paste the contents of a cell.

' Create and select objects.
[A:B2].Contents = "1"
[A:B2].Select
' Cut the selection.
CurrentDocument.CutSelection
' Paste the cut material into B3.
[A:B3].Paste
```

1-2-3: Cut method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DOCUMENT_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_CUT_METHOD_EXSCRIPT',1)} See example

Copies the object to the Clipboard and deletes it.

Syntax

object.**Cut** [*format*]

Parameters

format

(Optional) Variant (ClipboardFormat enumeration). The format in which to cut to the Clipboard. The following table lists the allowed values for this parameter.

Value	Description
$NativeFormat	Native format
$RichTextFormat	Rich text format
$BitmapFormat	Bitmap format
$PictureFormat	Picture format
$LotusChartFormat	Lotus Chart format
$EmbedSourceFormat	Embed source format
$EmbeddedObjectFormat	Embedded object format
$DIBFormat	Device-independent bitmap
$TextFormat	Text
$WK1Format	1-2-3 for DOS, Release 2
$WK3Format	1-2-3 for Windows Release 1; 1-2-3 for DOS Releases 3, 4

Return values

None

{button ,AL(`H_123_CUTSELECTION_METHOD_MEMDEF;H_123_PASTE_METHOD_MEMDEF',0)} See related topics

```
' Example: Cut and Paste methods
' Put "1" in B1, then cut it out and paste it in B2.
[A:B1].Contents = "1"
[A:B1].Cut
[A:B2].Paste
```

1-2-3: DataParseGuess method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DATAPARSEGUESS_METHOD_EXSCRIPT',1)} See example

Parses a range, returns a string of guessed data formats, and outputs a parsed format line. This method parses for labels, values, dates, and times. For more information, search on "Parsing data" in the 1-2-3 Help Index.

Syntax

format **=** *range*.**DataParseGuess**(*destinationrange*)

Parameters

destinationrange

Range. The output range for the parsed data format.

Return values

String. The guessed data formats, including the types of data and their block lengths.

{button ,AL(`H_123_DATAPARSE_METHOD_MEMDEF',0)} See related topics

```
' Example: CellValue property; DataParseGuess and DataParse methods
' Guess the data format for a cell containing a text block and a numeric block.
' Then parse the cell using the guessed format.
[B2].Contents = "AAA 123"
[B3].Contents =  [B2].DataParseGuess
[B2].DataParse [B4], [B3].CellValue
```

1-2-3: DataParse method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DATAPARSE_METHOD_EXSCRIPT',1)} See example

Parses a range containing text, using specified data formats. Puts the extracted data into the specified destination range, one column per block of data.

Syntax

range.**DataParse** *destinationrange, formatline*

Parameters

destinationrange

Variant. The output range for the parsed data.

formatline

String. The data formats to use to parse the data. You can parse for numbers, labels, dates, and times in blocks of specified length. For more information, search on "Format line" in the 1-2-3 Help Index.

Return values

None

Usage

The source range object on which you run this method may be read-only or read-write. The destination range must be read-write.

{button ,AL(`H_123_DATAPARSEGUESS_METHOD_MEMDEF',0)} See related topics

```
' Example: DataParseGuess, DataParse methods
' Declare a variable to hold the format line for parsing.
Dim parseString As String

' Set up a label range to be parsed, containing
' two label blocks and one date block.
[A:B3].Contents = "Smith  Arthur   5/23/89"
[A:B4].Contents = "Aubry  Lisa   4/12/87"
[A:B5].Contents = "Howard  Janet   2/6/84"

' Guess the data format based upon the labels.
parseString = [A:B3..A:B5].DataParseGuess
' Parse the range using the guessed format
' and put the resulting data in a range starting at B10.
[A:B3..A:B5].DataParse [A:B10],parseString

' Output:
' A: --- B --------- C ----- D --
'  10   Smith      Arthur 05/23/89
'  11   Aubry      Lisa   04/12/87
'  12   Howard     Janet  02/06/84
```

1-2-3: DefineNamedStyle method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFINENAMEDSTYLE_METHOD_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT', 1)} See example

Defines a named style based on the styles of the source cell, which is the top left cell of this range.

Syntax

range.**DefineNamedStyle** *stylename*

Parameters

stylename

String. The name for the named style.

Return values

None

{button ,AL(`H_123_DELETENAMEDSTYLE_METHOD_MEMDEF;H_123_MODIFYNAMEDSTYLE_METHOD_MEMDEF;H_123_RENAMENAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTONAMEDSTYLE_METHOD_MEMDEF;H_123_STYLE_PROPERTY_MEMDEF;',0)} See related topics

```
' Example: Bold, Font, and StyleName properties;
'          DefineNamedStyle and DeleteNamedStyle methods
' Define a named style, use it, and delete it.
[A:B2].Contents = "Louise"
[A:C2].Contents = "Norman"
' Set the style for text in a cell to bold and name the style.
[A:B2].Font.Bold = True
' Name this style as BoldText.
[A:B2].DefineNamedStyle "BoldText"
' Apply this style to another range.
[A:C2].StyleName = "BoldText"
' When finished with the named style, you can delete it.
.DeleteNamedStyle "BoldText"
```

1-2-3: DeleteColumns method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETECOLUMNS_METHOD_EXSCRIPT',1)} See example

Deletes the columns in the specified range.

Syntax

range.**DeleteColumns** [*deletetype*]

Return values

None

Parameters

deletetype

(Optional) Variant (ColRowType enumeration). Determines what part of the columns to delete. The following table lists the allowed values for this parameter.

Value	Description
$Full	Delete the entire column.
$Partial	Delete only the part of the column that is inside the range.

{button ,AL(`H_123_DELETEROWS_METHOD_MEMDEF;H_123_DELETESHEET_METHOD_MEMDEF;H_123_INSERTROWS_METHOD_MEMDEF;H_123_INSERTSHEET_METHOD_MEMDEF;H_123_NEWSHEET_METHOD_MEMDEF',0)} See related topics

```
' Example: DeleteColumns method
' Delete the full column.
[A:B2].DeleteColumns $Full
```

1-2-3: DeleteComputedField method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_CREATECOMPUTEDFIELD_METHOD_EXSCRIPT',1)} See example

Deletes a computed field from a query.

Syntax

dataquery.**DeleteComputedField** *formula*

Parameters

formula

String. The formula string or field name of the computed field that you want to delete from the query.

Return values

None

{button ,AL(`H_123_CREATECOMPUTEDFIELD_METHOD_MEMDEF;H_123_ADDSELECTFIELD_METHOD_MEMDEF',0)} See related topics

1-2-3: DeleteNamedPrintSettings method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_DELETENAMEDPRINTSETTINGS_METHOD_EXSCRIPT',1)} See example

Deletes the given print settings name from the document's NamedPrintSettings collection.

Syntax

document.**DeleteNamedPrintSettings** *settingsname*

Parameters

settingsname

String. The print settings name to be deleted from the NamedPrintSettings collection.

Return values

None

{button ,AL(`H_123_NEWNAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NAMEDPRINTSETTINGS_PROPERTY_MEMDEF;H_123_USEDEFAULTPRINTSETTINGS_METHOD_MEMDEF;H_123_CURRENTPRINTSETTINGS_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CurrentPrintSettings property; DeleteNamedPrintSettings,
'          NewNamedPrintSettings, Print, UseDefaultPrintSettings, methods
' This example creates a named print settings, uses it, then deletes it.
' First, create a new named print settings.
Dim myPrtSet As PrintSettings
Set myPrtSet = CurrentDocument.NewNamedPrintSettings("Prt_Set_1", False, )
' Do something with the print settings. For example, set a margin and footer.
myPrtSet.LeftMargin = 1440      ' In units of twips
myPrtSet.FooterLeft = "^"       ' Print file name
' Make the current document use Prt_Set_1.
Set CurrentDocument.CurrentPrintSettings = [Prt_Set_1]
' Print the current document.
CurrentApplication.Print
' Continue using the print settings as needed ...
' Optionally, go back to the application's default print settings.
CurrentDocument.UseDefaultPrintSettings
' If necessary, delete the print settings name when finished.
CurrentDocument.DeleteNamedPrintSettings "Prt_Set_1"
```

1-2-3: DeleteNamedStyle method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFINENAMEDSTYLE_METHOD_EXSCRIPT',1)} See example

Deletes the named style. Does not change the style of any ranges having this named style.

Syntax

range.**DeleteNamedStyle** *stylename*

Parameters

stylename

String. The name of the named style.

Return values

None

{button ,AL(`H_123_DEFINENAMEDSTYLE_METHOD_MEMDEF;H_123_MODIFYNAMEDSTYLE_METHOD_MEMDEF;H_123_RENAMENAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTONAMEDSTYLE_METHOD_MEMDEF;H_123_STYLENAME_PROPERTY_MEMDEF;',0)} See related topics

1-2-3: DeleteQuery method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CREATECOMPUTEDFIELD_METHOD_EXSCRIPT',1)} See example

Deletes the specified query from the document.

Syntax

document.**DeleteQuery** *queryname*

Parameters

queryname

String. The name of the query to delete.

Return values

None

{button ,AL(`H_123_NEWQUERY_METHOD_MEMDEF;H_123_ADDSELECTFIELD_METHOD_MEMDEF',0)} See related topics

1-2-3: DeleteRangeName method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETERANGENAME_METHOD_EXSCRIPT',1)} See example

Deletes the specified range name from the document.

Syntax

document.**DeleteRangeName** *rangename*

Parameters

rangename

String. The range name to be deleted.

Return values

None

{button ,AL(`H_123_CLEARRANGENAMES_METHOD_MEMDEF;H_123_CREATERANGENAME_METHOD_MEMDEF;H_123_CREATERANGENAMEFROMLABEL_METHOD_MEMDEF;H_123_CREATERANGENAMETABLE_METHOD_MEMDEF;H_123_ISRANGENAMED_METHOD_MEMDEF;H_123_NAME_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CreateRangeName and DeleteRangeName methods, ColumnWidth property
' Create a range name, use it, and delete it.
' Create the range name "North Sales" for the range A:C10..A:C20.
CurrentDocument.CreateRangeName "North_Sales", [A:C10..A:C20]
' Use this named range. For example, set its column width.
[North_Sales].ColumnWidth = 10
' If necessary, delete the named range when finished.
CurrentDocument.DeleteRangeName "North_Sales"
```

1-2-3: DeleteRecords method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETERECORDS_METHOD_EXSCRIPT',1)} See example

From a database table, deletes the records that meet the specified criteria formula.

Syntax

range.**DeleteRecords** *criteria*

Parameters

criteria

String. The criteria formula. For more information, search on "Criteria" in the 1-2-3 Help Index.

Return values

None

{button ,AL(`H_123_APPENDRECORDS_METHOD_MEMDEF;H_123_FINDRECORDS_METHOD_MEMDEF;H_123_CREATETABLE_METHOD_MEMDEF;H_123_COMMIT_METHOD_MEMDEF',0)} See related topics

```
' Example: CreateRange, and DeleteRecords methods
Sub DeleteRecordsMethod
   ' Create a small range of data.
   Msgbox "Create a small range of data."
   [A:A1].Contents = "Name"
   [A:B1].Contents = "State"
   [A:A2].Contents = "Sally"
   [A:B2].Contents = "NY"
   [A:A3].Contents = "Aidan"
   [A:B3].Contents = "MA"
   [A:A4].Contents = "Siobhan"
   [A:B4].Contents = "CT"
   [A:A5].Contents = "Lester"
   [A:B5].Contents = "MA"
   ' Name the range SOURCEINPUT.
   CurrentDocument.CreateRangeName "sourceinput",[A:A1..A:B5]

   ' Delete records in the range that match the criteria "State" = "MA".
   Msgbox "Delete records matching STATE = MA."
   [SOURCEINPUT].DeleteRecords "State=""MA"""
End Sub
```

1-2-3: DeleteRows method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETEROWS_METHOD_EXSCRIPT',1)} See example

Deletes the rows in the specified range.

Syntax

range.**DeleteRows** [*deletetype*]

Parameters

deletetype

(Optional) Variant (ColRowType enumeration). Determines what part of the rows to delete. The following table lists the allowed values for this parameter.

Value	Description
$Full	Delete the entire row.
$Partial	Delete only the part of the row that is inside the range.

Return values

None

{button ,AL(`H_123_DELETECOLUMNS_METHOD_MEMDEF;H_123_DELETESHEET_METHOD_MEMDEF;H_123_HIDEROWS_METHOD_MEMDEF;H_123_INSERTCOLUMNS_METHOD_MEMDEF;H_123_INSERTROWS_METHOD_MEMDEF;H_123_NEWSHEET_METHOD_MEMDEF',0)} See related topics

```
' Example: DeleteRows method
' Delete the cells in A:B2..C3 and move the remaining data up to fill these cells.
[A:B2..A:C3].DeleteRows $Partial
```

1-2-3: DeleteSheet method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETESHEET_METHOD_EXSCRIPT',1)} See example

Deletes the sheet or collection of sheets.

Syntax

sheet.**DeleteSheet**

Parameters

None

Return values

None

{button ,AL(`H_123_DELETEROWS_METHOD_MEMDEF;H_123_DELETECOLUMNS_METHOD_MEMDEF;H_123_NEWSHEET_METHOD_MEMDEF',0)} See related topics

```
' Example: AddToSelection, DeleteSheet and NewSheet methods
' Delete sheets from a document in various ways: direct, selected, and named.
' First, create a new document.
CurrentApplication.NewDocument
' Add 12 sheets to the document.
CurrentDocument.NewSheet $Before, 12, True

' Delete sheet B.
Msgbox "Delete sheet B."
[B].DeleteSheet

' Delete the selected sheet.
Msgbox "Select sheet G and delete it."
[G].Select
.DeleteSheet

' Delete contiguous sheets.
Msgbox "Delete the contiguous sheets H and I."
[H].Select
[I].AddToSelection
Selection.DeleteSheet

' Delete a named sheet.
Msgbox "Name a sheet as "SheetB" and then delete it by name."
[B].Name = "SheetB"
[SheetB].DeleteSheet
```

1-2-3: DeleteVersionGroup method

{button ,AL(`H_123_VERSIONGROUP_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDVERSION_METHOD_EXSCRIPT',1)} See example

Deletes this VersionGroup object from the document.

Syntax

versiongroup.**DeleteVersionGroup**

Parameters

None

Return values

None

{button ,AL(`H_123_ADDVERSION_METHOD_MEMDEF;H_123_MERGEVERSIONS_METHOD_MEMDEF;H_123_NEWVERSIONGROUP_METHOD_MEMDEF;H_123_NEWVERSION_METHOD_MEMDEF;H_123_REMOVEVERSION_METHOD_MEMDEF;H_123_REPORTVERSION_METHOD_MEMDEF;H_123_VERSIONGROUP_METHOD_MEMDEF;H_123_VERSIONS_METHOD_MEMDEF;H_123_VERSION_METHOD_MEMDEF;H_123_CURRENTVERSION_PROPERTY_MEMDEF;H_123_LASTVERSIONGROUP_PROPERTY_MEMDEF;H_123_SHOWVERSIONBORDERS_PROPERTY_MEMDEF;H_123_VERSIONBORDERSVISIBLE_PROPERTY_MEMDEF;H_123_VERSIONBORDERVISIBLE_PROPERTY_MEMDEF',0)} See related topics

1-2-3: DeleteVersion method

{button ,AL(`H_123_VERSION_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETEVERSION_METHOD_EXSCRIPT',1)} See example

Deletes this Version object from its range.

Syntax

version.**DeleteVersion**

Parameters

None

Return values

None

{button ,AL(`H_123_NEWVERSION_METHOD_MEMDEF;H_123_ADDVERSION_METHOD_MEMDEF;H_123_MERGEVERSIONS_METHOD_MEMDEF;H_123_REMOVEVERSION_METHOD_MEMDEF;H_123_REPORTVERSION_METHOD_MEMDEF;H_123_VERSIONS_METHOD_MEMDEF;H_123_VERSION_METHOD_MEMDEF;H_123_CURRENTVERSION_PROPERTY_MEMDEF;H_123_SHOWVERSIONBORDERS_PROPERTY_MEMDEF;H_123_VERSIONBORDERSVISIBLE_PROPERTY_MEMDEF;H_123_VERSIONBORDERVISIBLE_PROPERTY_MEMDEF',0)} See related topics

```
' Example: DeleteVersion, NewVersion, CreateRangeName methods
' Create, use, and delete a range version.

' First, create some data for a range.
[A:B2].Contents = "10"
[A:B3].Contents = "20"
[A:B4].Contents = "30"

' Create a version named "Projected_Sales" of the
' range named "Q1Sales" and show the border.
CurrentDocument.CreateRangeName "Q1Sales", [A:B2..A:B4]
[Q1Sales].NewVersion("Projected_Sales")
[Q1Sales].VersionBorderVisible = True

' Use this version to do something. For example, increase all its values 10%.
ForAll sglcell In [Q1Sales].Cells
   sglcell.Contents = CStr(sglcell.CellValue * 1.1)
End Forall

' If necessary, delete the "Projected_Sales" version when finished.
If 6 = Messagebox("Delete the version?", 4 + 32) Then
   [Q1Sales.Projected_Sales].DeleteVersion
End If
```

1-2-3: DemoteColumn method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_COLLAPSECOLUMN_METHOD_EXSCRIPT',1)} See example

Demotes the outline level of the range by the specified number of levels. You can run this method on the currently selected range or on a range object reference.

Syntax

range.**DemoteColumn** [*levels*]

Parameters

levels

(Optional) Long. The number of levels to demote from the current level. The default is one level.

Return values

None

Usage

If demotion of a column would make the difference in levels between it and its parent (summary column) more than one level, this method doesn't demote the column.

If a column would hit the maximum demotion level, this method doesn't demote the column. If some columns in the range can be demoted, while others have reached the maximum level or can't move further below their parents, the columns that can move are demoted.

This method works on collections of ranges, but doesn't work on 3D ranges.

The default outline folding is parent before children, but you can reverse this by setting the Sheet.ColumnFolding property to $ParentAfter.

{button ,AL(`H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF,H_123_COLLAPSECOLUMN_METHOD_MEMDEF,H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_COLUMNFOLDING_PROPERTY_MEMDEF',0)} See related topics

1-2-3: DemoteRow method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DEMOTEROW_METHOD_EXSCRIPT',1)} See example

Demotes the outline level of the range by the specified number of levels. You can run this method on the currently selected range or on a range object reference.

Syntax

range.**DemoteRow** [*levels*]

Parameters

levels

(Optional) Long. The number of levels to demote from the current level. The default is one level.

Return values

None

Usage

If demotion of a row would make the difference in levels between a parent (summary row) and child more than one level, this method doesn't demote the row.

If a row would hit the maximum demotion level, this method doesn't demote the row. If some rows in the range can be demoted, while others have reached the maximum level or can't move further below the parent, the rows that can move are demoted.

This method works on range collections, but not on 3D ranges.

The default outline row folding is parent after children, but you can reverse this by setting the Sheet.RowFolding property to $ParentBefore.

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_ROWFOLDING_PROPERTY_MEMDEF',0)} See related topics

```
' Example: DemoteRow, Select, and AddToSelection methods
' Demote rows 2 through 10 in sheet A.
[A:A2..A:A10].DemoteRow
' Demote the rows in the currently selected range.
[].DemoteRow
' Select a collection of rows (3 and 5) and demote them.
[A:A3..A:IV3].Select
[A:A5..A:IV5].AddToSelection
Selection.DemoteRow
```

1-2-3: DisableItem method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

{button ,AL(`H_123_GETMENU_METHOD_EXSCRIPT',1)} See example

Disables an item in a menu or menu bar.

Syntax

object.**DisableItem** *position*

Parameters

position

Long. The menu position of the item to be disabled.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position.

Return values

None

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_ADDSEPARATOR_METHOD_MEMDEF;H_123_CHECKITEM_METHOD_MEMDEF;H_123_ENABLEITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_UNCHECKITEM_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF',0)} See related topics

1-2-3: Disconnect method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Disconnects from the specified external database range, ending all data exchange between 1-2-3 and the external table.

Syntax

document.**Disconnect** *externalrangename*

Parameters

externalrangename

String. The name of the external range from which you want to disconnect.

Return values

None

{button ,AL(`H_123_CONNECT_METHOD_MEMDEF;H_123_CREATETABLE_METHOD_MEMDEF;H_123_NEWQUERYTABLE_METHOD_MEMDEF;H_123_ROLLBACK_METHOD_MEMDEF',0)} See related topics

1-2-3: Distribution method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_DISTRIBUTION_METHOD_EXSCRIPT',1)} See example

Determines the distribution of values in a range and enters the distribution in an adjacent range.

Syntax

*document.***Distribution** *valuesrange, binrange*

Parameters

valuesrange

Variant. The range to scan for values. This range may be read-only or read-write. Text and blanks are ignored.

binrange

Variant. The range containing the list of bin values over which the input range will be distributed.

Return values

None

Usage

The output range appears in the column immediately to the right of the bin range. Each output cell contains the number of values in *valuesrange* that fall in the bin determined by the adjacent bin values.

```
' Example: CreateRangeName and Distribution methods
' Determine how sales were distributed for the month of June.
' Use the range named June, containing June sales data, as the values range.
Dim valuesRange As Range
Set valuesRange = Bind("A2..A10")
CurrentDocument.CreateRangeName "June", valuesRange
[A1].Contents = "Values range"
[A2].Contents = "1000"
[A3].Contents = "2500"
[A4].Contents = "5000"
[A5].Contents = "2200"
[A6].Contents = "3000"
[A7].Contents = "3200"
' Use [B2..B4] as the bin range.
[B1].Contents = "Bin range"
[B2].Contents = "2000"
[B3].Contents = "3000"
[B4].Contents = "4000"
[C1].Contents = "Output"
' Calculate the distribution.
CurrentDocument.Distribution [June], [B2..B4]
' Output appears in cells [C2..C5].
' C2 is 1 (there is  1 value  in Values range below 2000).
' C3 is 3 (there are 3 values in Values range between 2000 and 3000).
' C4 is 1 (there is  1 value  in Values range between 3000 and 4000).
' C5 is 1 (there is  1 value  in Values range above 4000).
```

1-2-3: DragAndFill method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_DRAGANDFILL_METHOD_EXSCRIPT',1)} See example

Fills a specified target range with a sequence of data that follows a pattern based on data already entered in the source *range* that you run this method on. See 1-2-3 Help Index on "Filling ranges" for details on the data pattern that 1-2-3 creates.

Syntax

range.**DragAndFill** *fillrange*

Parameters

fillrange

Range. The 2D target range to be filled with a sequence of data. The fill range must completely overlap the source *range* that you run this method on, and must start on the same root cell as the source range.

Return values

None

Usage

Neither the source range nor the fill range can be a 3D range.

If the source range has more than one cell, the fill proceeds only in one direction, along rows or along columns. Only if the source range is a single cell can the fill proceed in both directions.

If the fill range is smaller than the source range, no fill occurs, but the part of the source range outside the fill range is cleared of data.

{button ,AL(`H_123_RANGEFILL_METHOD_MEMDEF;H_123_FILLRANGE_PROPERTY_MEMDEF;H_123_RANGEFILLINTERVAL_PROPERTY_MEMDEF;H_123_RANGEFILLSTART_PROPERTY_MEMDEF;H_123_RANGEFILLSTEP_PROPERTY_MEMDEF;H_123_RANGEFILLSTOP_PROPERTY_MEMDEF;H_123_RANGEFILLTYPE_PROPERTY_MEMDEF;',0)} See related topics

```
' Example: DragAndFill method
' Example 1: Fill a range with month names.
' Fill all 12 months in cells A1 to A12.
[A1].Contents = "January"
[A1].DragAndFill [A1..L1]
' Output: Row 1 now has the months of the year in the first 12 columns.

' Example 2: Fill a range with one name and sequential numbers.
[A2].Contents = "Item 1"
[A3].Contents = "Item 2"
' Fill cells A4..A10 with "Item n".
[A2..A3].DragAndFill [A2..A10]
' Output: Column A now has "Item 3" in A4, and so on
'         through "Item 9" in A10.
```

1-2-3: EnableItem method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

{button ,AL(`H_123_ENABLEITEM_METHOD_EXSCRIPT',1)} See example

Enables an item in a menu or menu bar.

Syntax

object.**EnableItem** *position*

Parameters

position

Long. The menu position of the item to be enabled.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position.

Return values

None

Usage

This method resets a 1-2-3 default menu item to its state, enabled or disabled, prior to the running of the script. You can't enable a 1-2-3 default menu item that 1-2-3 has disabled, but you can enable one that you have disabled.

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_ADDSEPARATOR_METHOD_MEMDEF;H_123_CHECKITEM_METHOD_MEMDEF;H_123_DISABLEITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_UNCHECKITEM_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: EnableItem, GetMenu, GetMenuPosition, and GetItemText methods
' Get the positions of the Window menu and the New Window menu item, and
' enable the New Window item.
Dim menu1 As Menu                   ' Window menu object
Dim position As Long                ' Position of Window menu in menu bar
' Get the Window menu object.
position = CurrentApplication.GetMenuPosition($WindowMenu)
Set menu1 = CurrentApplication.CurrentMenuBar.GetMenu(position)
' Verify the position of the New Window item and enable it.
If menu1.GetItemText(1) = "&New Window" Then
   menu1.EnableItem 1
Else
   ' The New Window item has been displaced to a different position.
   ' Loop over the other items to find the New Window item and then enable it ...
End If
```

1-2-3: EndPoll method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTPOLL_METHOD_EXSCRIPT',1)} See example

Terminates the specified poll event. If the poll event is operating, it ceases operating, and no further events will be raised. If the poll event is not operating, nothing happens.

Syntax

document.**EndPoll** *eventindex*

Parameters

eventindex

Long. Index of the event to be terminated. A value of 1 for *eventindex* specifies the event Poll1, and so on through the event Poll4.

Return values

None

{button ,AL(`H_123_STARTPOLL_METHOD_MEMDEF;H_123_POLL1_EVENT_MEMDEF;H_123_POLL2_EVENT_MEMDEF;H_123_POLL3_EVENT_MEMDEF;H_123_POLL4_EVENT_MEMDEF',0)} See related topics

1-2-3: ExpandColumn method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_COLLAPSECOLUMN_METHOD_EXSCRIPT',1)} See example

Expands collapsed columns in the range.

Syntax

range.**ExpandColumn**

Parameters

None

Return values

None

Usage

This method doesn't work on 3D ranges.

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_COLLAPSEALL_METHOD_MEMDEF;H_123_EXPANDALL_METHOD_MEMDEF',0)} See related topics

1-2-3: ExpandRow method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_COLLAPSEROW_METHOD_EXSCRIPT',1)} See example

Expands collapsed rows in the range.

Syntax

range.**ExpandRow**

Parameters

None

Return values

None

Usage

This method doesn't work on 3D ranges.

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_COLLAPSEALL_METHOD_MEMDEF;H_123_EXPANDALL_METHOD_MEMDEF',0)} See related topics

1-2-3: ExtendedName method

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS', 0)} See list of classes

{button ,AL(`H_123_EXTENDEDNAME_METHOD_EXSCRIPT',1)} See example

Returns the name of the object, including the specified parts of the name.

Syntax

extendedname = *object*.**ExtendedName**(*nametype*)

Parameters

nametype

Variant (enumeration). The parts of the extended name to return. The following table lists the allowed values for this parameter. The values are from the ExtendedNameType enumeration.

Value	Description
$ShortName	By default, the object name. For a Document object, the file name without path information. For a Version object, the VersionName property (for example, "Version 1").
$DocumentRelativeName	The name with no document qualification. For a Version object, the Name property (for example, "Range 1.Version 1").
$FullName	The name with document qualification. Generally for objects other than documents, the FullName is DocumentName & "." & ShortName. However, for a Document object, the FullName is DocumentName & "\" & ShortName. An example of the FullName of a Version object is "<<D:\lotus\work\123\file1.123>>Range 1.Version 1".
$DocumentName	The name of the object container (for example, "<<D:\lotus\work\123\file1.123>>"). For a Document object, the document file name.

Return values
String. The extended name of the object.

{button ,AL(`H_123_FULLNAME_PROPERTY_MEMDEF;H_123_NAME_PROPERTY_MEMDEF',0)} See related topics

```
' Example: ExtendedName method
' Return the filename of the current document,
' including the path information.
Dim filePath As String
filePath = CurrentDocument.ExtendedName($Fullname)
```

```
' Example: ExtendSelection, Select methods
' Extend a sheet selection forward.
' First, add 3 sheets to the current document.
CurrentDocument.NewSheet $After, 3, True
' Select sheet A and then extend the selection to include sheet B.
[A].Select
[B].ExtendSelection $Forward
```

1-2-3: ExtendSelection method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_EXTENDSELECTION_METHOD_EXSCRIPT',1)} See example

Extends the selection to this sheet. This method extends either a range or a sheet selection to include this sheet.

Syntax

sheet.**ExtendSelection**

Parameters

None

Return values

None

{button ,AL(`H_123_CLEARSELECTION_METHOD_MEMDEF;H_123_COPYSELECTION_METHOD_MEMDEF;H_123_CUTSELECTION_METHOD_MEMDEF;H_123_EXTENDSELECTFROMTAB_METHOD_MEMDEF;H_123_EXTENDWORKSHEETSELECTIONBACK_METHOD_MEMDEF;H_123_EXTENDWORKSHEETSELECTIONFORWARD_METHOD_MEMDEF;H_123_GETSELECTION_METHOD_MEMDEF;H_123_REMOVEFROMSELECTION_METHOD_MEMDEF;H_123_SELECT_METHOD_MEMDEF;H_123_SELECTALL_METHOD_MEMDEF;H_123_SELECTALLSHEETS_METHOD_MEMDEF;H_123_ISSELECTABLE_PROPERTY_MEMDEF;H_123_ISSELECTED_PROPERTY_MEMDEF;H_123_SELECTION_PROPERTY_MEMDEF;H_123_SELECTED_EVENT_MEMDEF',0)} See related topics

```
' Example: ExtendSheetSelectionBack, Goto, and NewSheet methods
' Select a sheet and then extend the selection to include another sheet.

' First, create a new document.
CurrentApplication.NewDocument
' Add 3 sheets to the document.
CurrentDocument.NewSheet $After, 3, True
' Select cell D:B2.
[D:B2].Select
' Move the cell pointer also to cell D:B2,
' as required by the ExtendSheetSelectionBack method.
[D:B2].Goto
' Extend the selection to the previous sheet.
MessageBox "Extend the selection to the previous sheet."
CurrentDocument.ExtendSheetSelectionBack
```

```
' Example: ExtendSheetSelectionForward, Goto, and NewSheet methods
' Select a sheet and then extend the selection to include another sheet.

' First, create a new document.
CurrentApplication.NewDocument
' Add 6 sheets to the document.
CurrentDocument.NewSheet $After, 6, True

' Select cell D:B2.
[D:B2].Select
' Move the cell pointer also to cell D:B2,
' as required by the ExtendSheetSelectionBack method.
[D:B2].Goto
' Extend the selection to the next sheet.
Messagebox "Extend the selection to the next sheet."
CurrentDocument.ExtendSheetSelectionForward
```

1-2-3: ExtendSheetSelectionBack method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_EXTENDSHEETSELECTIONBACK_METHOD_EXSCRIPT',1)} See example

Extends the selected range, or entire sheet selection, to the previous sheet and goes to that sheet.

Syntax

document.**ExtendSheetSelectionBack**

Parameters

None

Return values

None

{button ,AL(`H_123_EXTENDWORKSHEETSELECTIONFORWARD_METHOD_MEMDEF;H_123_EXTENDSELECT_METHOD_MEMDEF;H_123_SELECTION_PROPERTY_MEMDEF',0)} See related topics

1-2-3: ExtendSheetSelectionForward method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_EXTENDSHEETSELECTIONFORWARD_METHOD_EXSCRIPT',1)} See example

Extends the selected range, or the entire sheet selection, to the next sheet and goes to that sheet.

Syntax

*document.***ExtendSheetSelectionForward**

Parameters

None

Return values

None

{button ,AL(`H_123_EXTENDWORKSHEETSELECTIONBACK_METHOD_MEMDEF;H_123_EXTENDSELECT_METHOD_MEMDEF;H_123_SELECT_METHOD_MEMDEF;H_123_SELECTION_PROPERTY_MEMDEF',0)} See related topics

1-2-3: FieldAggregateType method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_FIELDAGGREGATETYPE_METHOD_EXSCRIPT',1)} See example

Performs a calculation on a group of data from a query table by setting up an aggregate column.

Syntax

dataquery.**FieldAggregateType** *fieldname, function*

Parameters

fieldname

String. The name of the field to set as aggregate.

function

Variant (SummaryOp enumeration). The aggregate or summary operation to be performed on the field. The following table lists the allowed values for this parameter.

Value	Description
$Sum	Adds the values
$Average	Averages the values
$Avg	Averages the values
$Count	Counts the values
$Maximum	Finds the largest value
$Max	Finds the largest value
$Minimum	Finds the smallest value
$Min	Finds the smallest value
$Reset	Resets the values

Return values

None

{button ,AL(`H_123_NEWQUERY_METHOD_MEMDEF;H_123_CREATERANGENAME_METHOD_MEMDEF',0)} See related topics

```
' Example: FieldAggregateType, FieldAlias, GetFieldAlias,
'          NewQuery, and Refresh methods; OutputLocation property
' Set a database table field to be a sum field.

' First, set up the query.
[A:B2].Contents = "Employee"
[A:C2].Contents = "EmpID"
[A:D2].Contents = "AmountSold"
[A:B3].Contents = "1"
[B3].DragAndFill [B3..D5]
CurrentDocument.NewQuery "Query 1", "A:B2..A:D10"

' Run the query.
[Query 1].OutputLocation = "A:B13"
[Query 1].Refresh

' Set the "AmountSold" field to aggregate as summed.
[Query 1].FieldAggregateType "AmountSold", $SUM

' Set the field alias to reflect the new aggregate.
[Query 1].FieldAlias "AmountSold", "Total Amount Sold By All Employees"
[Query 1].Refresh

...

' Later, get the alias.
Dim alias1 As String
alias1 = [Query 1].GetFieldAlias("AmountSold")
```

1-2-3: FieldAlias method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_FIELDAGGREGATETYPE_METHOD_EXSCRIPT',1)} See example

Sets the alias of the given field. The alias is the name shown in the first row for the field in a query table.

Syntax

dataquery.**FieldAlias** *fieldname, alias*

Parameters

fieldname

String. The name of the field to be given an alias.

alias

String. The alias of the field.

Return values

None

{button ,AL(`H_123_CREATECOMPUTEDFIELD_METHOD_MEMDEF;H_123_GETFIELDALIAS_METHOD_MEMDEF;H_123_FIELDAGGREGATETYPE_METHOD_MEMDEF;H_123_JOIN_METHOD_MEMDEF',0)} See related topics

1-2-3: FileAdminLinksRefresh method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_FILEADMINLINKSREFRESH_METHOD_EXSCRIPT',1)} See example

Updates formulas in all open files that contain links to other files.

Syntax

application.**FileAdminLinksRefresh**

Parameters

None

Return values

None

{button ,AL(`H_123_UPDATE_METHOD_MEMDEF;H_123_LINKUPDATE_EVENT_MEMDEF;H_123_AUTOUPDATE_PROPERTY_MEMDEF;H_123_UPDATELINKSONOPENDOC_PROPERTY_MEMDEF;H_123_CALC_METHOD_MEMDEF;H_123_RECALCRANGE_METHOD_MEMDEF',0)} See related topics

```
' Example: FileAdminLinksRefresh method
' Update all linked formulas in open files.
CurrentApplication.FileAdminLinksRefresh
```

1-2-3: Find method

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_FIND_METHOD_EXSCRIPT',1)} See example

Finds a cell containing the string stored in the Application.SearchString property and selects it. You can find the *n*th occurrence by using the optional occurrence argument.

Syntax

boolean **=** *document.object*.**Find**[(*occurrence*)]

Parameters

occurrence

(Optional) Long. Specifies which occurrence of the target string to find. The default is 1.

Return values

Variant (Boolean). Indicates whether an occurrence of the target string was found (value True) or not found (value False).

Usage

This method searches for the string stored in the Application.SearchString property, and uses the search matching option properties also stored in the Application object.

{button ,AL(`H_123_SEARCHSTRING_PROPERTY_MEMDEF;H_123_SEARCHLABELS_PROPERTY_MEMDEF;H_123_SEARCHFORMULAS_PROPERTY_MEMDEF;H_123_SEARCHVALUES_PROPERTY_MEMDEF;H_123_MATCHCASE_PROPERTY_MEMDEF;H_123_MATCHACCENT_PROPERTY_MEMDEF;H_123_MATCHPITCH_PROPERTY_MEMDEF;H_123_MATCHKATAKANA_PROPERTY_MEMDEF;H_123_REPLACE_METHOD_MEMDEF;H_123_REPLACEALL_METHOD_MEMDEF',0)} See related topics

```
' Example: MatchCase, SearchLabels, and SearchString properties; Find method
' Find the second occurrence of "Stock Quote" in a label cell
' of the current document, that matches the case exactly.
Dim found As Variant
CurrentApplication.SearchString = "Stock Quote"
CurrentApplication.MatchCase = True
CurrentApplication.SearchLabels = True
found = CurrentDocument.Find(2)
```

1-2-3: FitTallest method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_FITTALLEST_METHOD_EXSCRIPT',1)} See example

Adjusts the height of each row in a range, or a collection of ranges, to fit the largest font in that row.

Syntax

range.**FitTallest**

Parameters

None

Return values

None

{button ,AL(`H_123_FITWIDEST_METHOD_MEMDEF;H_123_FITWIDESTNUMBER_METHOD_MEMDEF;H_123_FITROWHEIGHTTOFONT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: DefaultRowHeight, Font, and Size properties;
'          FitTallest and ResetRowHeight methods
' Set a large font size in B2 and adjust the row size to accommodate.

' First, set the default row height to 10 pt.
[A].DefaultRowHeight = 10

' Set the font size in B2 to 14 points.
' (The row height is automatically adjusted to fit.)
[A:B2].Contents = "Text"
[A:B2].Font.Size = 14

' Reset all the row heights to the default.
Msgbox "Reset all the row heights to the default (10)."
[A:B2].ResetRowHeight

' Readjust the size of row 2 to accommodate the larger font size in B2.
Msgbox "Readjust the size of row 2 to accommodate the height in B2."
[A:B2].FitTallest
```

1-2-3: FitWidestNumber method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_FITWIDESTNUMBER_METHOD_EXSCRIPT',1)} See example

Adjusts the column widths to fit the widest numeric entries included in a range or collection of ranges. This method ignores label cells in fitting the width.

Syntax

range.**FitWidestNumber**

Parameters

None

Return values

None

{button ,AL(`H_123_FITWIDEST_METHOD_MEMDEF;H_123_FITTALLEST_METHOD_MEMDEF;H_123_FITROWHEIGHTTOFONT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: FitWidestNumber method
' Put more digits in a cell and adjust the column width accordingly.

' Set up an example cell.
[A:B2].Contents = "102,003,000"
[A:B2].FitWidestNumber
' Multiply the cell by 10.
' General-purpose checking for numeric contents is not actually needed
' for this cell, but would be for unknown cells.
If IsNumeric([A:B2].Contents) Then
   [A:B2].Contents = CStr(CLng([A:B2].Contents) * 10)
End If
' Then adjust the column width to the larger number.
[A:B2].FitWidestNumber
```

1-2-3: FitWidest method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_FITWIDEST_METHOD_EXSCRIPT;H_123_FORMATDECIMALS_AND_FORMATNAME_PROPERTIES_EXSCRIPT ',1)} See example

Adjusts the column widths to fit the widest cell entries included in a range or collection of ranges.

Syntax

range.**FitWidest**

Parameters

None

Return values

None

{button ,AL(`H_123_FITTALLEST_METHOD_MEMDEF;H_123_FITWIDESTNUMBER_METHOD_MEMDEF;H_123_FITROWHEIGHTTOFONT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: FitWidest method
' Enter a long label in B2 and adjust the column size to accommodate it.
' New cell contents should be longer than any others in column B.
[A:B2].Contents = "Long text entry ................"
' Adjust the column size to accommodate the largest cell.
[A:B2].FitWidest
```

1-2-3: FlipLeftRight method

{button ,AL(`H_123_ARC_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_FLIPLEFTRIGHT_METHOD_EXSCRIPT',1)} See example

Flips the object from left to right.

Syntax

document.object.**FlipLeftRight**

Parameters

None

Return values

None

{button ,AL(`H_123_FLIPTOPBOTTOM_METHOD_MEMDEF;H_123_ROTATION_PROPERTY_MEMDEF',0)} See related topics

```
' Example: FlipLeftRight and NewArrow methods
' This example flips an arrow from pointing right to pointing left.
' Draw an arrow on sheet A pointing right.
Dim myArrow As DrawLine
Set myArrow = [A].NewArrow (1440, 1440, 2880, 1440)
MessageBox "FlipLeftRight", MB_OK + MB_ICONINFORMATION, "FlipLeftRight example"
' Flip the arrow horizontally.
myArrow.FlipLeftRight
' The arrow now points left.
```

1-2-3: FlipTopBottom method

{button ,AL(`H_123_ARC_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_FLIPTOPBOTTOM_METHOD_EXSCRIPT',1)} See example

Flips the object from top to bottom, leaving the bounding rectangle unchanged, so that the object appears upside down.

Syntax

document.object.**FlipTopBottom**

Parameters

None

Return values

None

{button ,AL(`H_123_FLIPLEFTRIGHT_METHOD_MEMDEF;H_123_ROTATION_PROPERTY_MEMDEF',0)} See related topics

```
' Example: AddPoint, FlipTopBottom, and NewPolygon methods
' This example flips a triangle from pointing down to pointing up.
' Draw a triangle on sheet A pointing down.
Dim myPolygon As Polygon
' Begin by creating a Polygon object and drawing the first line segment.
Set myPolygon = [A].NewPolygon (1440, 1440, 2160, 2880)
' Draw the second line segment.
myPolygon.AddPoint 2880, 1440
' 1-2-3 automatically draws the third line segment,
' by connecting the first point to the last.
' Flip the triangle vertically.
Messagebox "FlipTopBottom.", MB_OK + MB_ICONINFORMATION, "FlipTopBottom example"
myPolygon.FlipTopBottom
' The triangle now points up.
```

1-2-3: FormatReset method

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_FORMATRESET_METHOD_EXSCRIPT',1)} See example

Resets a range to the sheet default number format.

Syntax

object.**FormatReset**

Parameters

None

Return values

None

{button ,AL(`H_123_FORMAT_METHOD_MEMDEF;H_123_ISVALIDFORMAT_METHOD_MEMDEF;H_123_FORMATDECIMALS_PROPERTY_MEMDEF;H_123_FORMATNAME_PROPERTY_MEMDEF',0)} See related topics

```
' Example: FormatReset method, FormatName property
' Format the cell B2 as US currency and then reset it to the default format.
[A:B2].FormatName = "US Dollar"
' Use this format for a while ...
' Then reset it to the default.
[A:B2].FormatReset
```

1-2-3: Format method

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_FORMAT_METHOD_EXSCRIPT',1)} See example

Sets the number format and the number of decimal places for a range or sheet.

Syntax

range.**Format** *formatname,* [*formatdecimal*]

Parameters

formatname

String. The name of the format as it appears in the Current Format list on the Number Format tab in the InfoBox for ranges or sheets.

formatdecimal

(Optional) Long. The number of digits displayed to the right of the decimal point. Valid numbers are from 0 to 15.

Return values

None

Usage

This method is equivalent to setting the FormatName and FormatDecimals property.

{button ,AL(`H_123_FORMATRESET_METHOD_MEMDEF;H_123_ISVALIDFORMAT_METHOD_MEMDEF;H_123_FORMATDECIMALS_PROPERTY_MEMDEF;H_123_FORMATNAME_PROPERTY_MEMDEF;H_123_CENTURYLONGFORMAT_PROPERTY_MEMDEF;H_123_ISFORMATFREQUSED_PROPERTY_MEMDEF;',0)} See related topics

```
' Example: Format method
' Format a range in Fixed format and then in Scientific format.
' First, set up a range with some data.
Dim myRange As Range

Set myRange = Bind("A:C10..A:C20")
[A:C10].Contents = "1234.5678"
' Format the range in Fixed format with two decimal places.
myrange.Format "Fixed", 2
' Result: cell C10 displays "1234.57".
' Format the range in Scientific format with two decimal places.
myrange.Format "Scientific", 2
' Result: cell C10 displays "1.23E+003".
```

1-2-3: FreeCellData method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_FREECELLDATA_METHOD_EXSCRIPT',1)} See example

Frees the memory buffer allocated in a previous call to the GetCellData method.

Syntax

range.**FreeCellData** *celldatapointer*

Parameters

celldatapointer

Long. A pointer to an array of pointers that was returned by the GetCellData method.

Return values

None

{button ,AL(`H_123_GETCELLDATA_METHOD_MEMDEF;H_123_CLEAR_METHOD_MEMDEF;H_123_CONTENTS_PROPERTY_MEMDEF',0)} See related topics

```
' Example: GetCellData, FreeCellData and SetCellData methods
' Get an array of data from the selected range,
' call a C library function that transposes the array,
' and then free the data buffer.

' To run the example:
'
' 1.  Build the C++ source code given below (following the script code)
'     into a DLL. No other files are needed in the project. The DLL exports
'     a function TransposeDoub that transposes an array of doubles.
' 2.  In the Global declarations, change the Declare statements to point to
'     the proper path and DLL name.
' 3.  In a 1-2-3 worksheet, create and select a range with at least 3 rows,
'     filled with doubles.
' 4.  Run TransposeDoub to transpose the range of doubles.
'     The transposition is described in the C++ source code.

' Global declarations for the script
Option Public

' Define the DLL worker function.
Declare Function TransposeDoub Lib "\proj\chngcell\debug\chngcell.dll" _
(Byval ptr As Long, Byval rows As Long, Byval cols As Long) As Long

' The following sub transposes an array of doubles.

Sub Click(Source As ButtonControl)
   Dim inRange As Range          ' The range containing the original data
   Dim arrayPtr As Long          ' Pointer to the array of data pointers
   Dim rows As Long              ' Number of rows
   Dim cols As Long              ' Number of columns

   ' Get the number of rows and columns.
   Set inRange = Selection
   rows = inRange.EndRow  - inRange.StartRow  + 1
   cols = inRange.EndColumn - inRange.StartColumn + 1
   Print inRange.Name

   ' Transpose the array.
   arrayPtr = InRange.GetCellData($Double)
   Print TransposeDoub(arrayPtr, rows, cols)
   inRange.SetCellData arrayPtr
   Print "Done with set."

   ' Free the data buffer.
   inRange.FreeCellData arrayPtr

End Sub

' The following is the C++ source code for the transpose routine.

celld.c

#include <malloc.h>
#define DllExport __declspec( dllexport )
//
// Cell array header
//
```

```
typedef struct _CellDataHdr {
   unsigned short hdrSize;            // Length of this struct
   unsigned short IsDouble;           // 1 if the following array is double
                                      // 0 if it is vector of char *-s
   unsigned long size;                // Number of elements in the array
   unsigned long rows, cols, sheets;  // Number of rows, columns, and sheets
                                      // Set by GetCellData method... ignored
                                      // Not used by other methods
} CellDataHdr, *PCellDataHdr;

// The cell array transposition follows this pattern:
//
//      From ---------------->To
//
//      <--cols-->           <--cols-->
//  ^   A  B  C  D       ^   Q  R  S  T
//  |   E  F  G  H       |   M  N  O  P
// rows I  J  K  L  ==> rows I  J  K  L
//  |   M  N  O  P       |   E  F  G  H
//  v   Q  R  S  T       v   A  B  C  D
//

//
// Transpose a cell array of doubles.
//
DllExport long TransposeDoub(unsigned long ptr, unsigned long rows, unsigned long
cols)
{
   PCellDataHdr hdr = (PCellDataHdr) ptr;
   double *buf = (double *) &hdr[1];
   unsigned int i, j;
   int count = 0;

   // Validate arguments and table.
   if (rows <= 2 ||
      !hdr->IsDouble ||
      rows > hdr->rows ||
      cols > hdr->cols) {
      return 0;
      }

   for (i = 0; i < cols; i++) {
      double tmp, *pCol;
      pCol = &buf[i * hdr->rows];             // Calc column pointer.

      // Swap column rows.
      for (j = 0; j < rows/2; j++) {
         tmp = pCol[rows - j - 1];
         pCol[rows - j - 1] = pCol[j];
         pCol[j] = tmp;
         count++;
         }
      }
   return count;
}
```

1-2-3: GetCellData method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_FREECELLDATA_METHOD_EXSCRIPT',1)} See example

Gets the data in a range, referenced as a set of pointers to the contents of individual cells. This method allocates memory for copying the contents of the range, sets the values in memory, and returns an array pointer that can be used by an external C program.

Syntax

arraypointer = *range*.**GetCellData**(*celldatatype*)

Parameters

celldatatype

Variant (CellDataType enumeration). The format for the returned cell data. The following table lists the allowed values for this parameter.

Value	Description
$CellValue	Return a pointer to an array of strings. The strings contain the formatted values that the cells evaluate to. Blank cells result in NULL pointers.
$FormulaContents	Return a pointer to an array of strings. This is similar to $CellValue, except the strings are the contents of formula cells (for example, "+A1+@SUM(B1)"). All other types of cells result in NULL pointers.
$Double	Return a pointer to an array of doubles. Blank cells and labels are returned as zero. NA and ERR are encoded as invalid numbers.

Return values

Long. A pointer to an array of pointers, one for each cell in the range. The array is ordered by row/column/sheet. For example, an array representing the range [A:A1..B:B2] would be given in the order A:A1, A:A2, A:B1, A:B2, B:A1, B:A2, B:B1, B:B2.

Usage

This method allocates the memory for the returned array internally, and the caller must free that memory.

{button ,AL(`H_123_FREECELLDATA_METHOD_MEMDEF;H_123_SETCELLDATA_METHOD_MEMDEF;H_123_CONTENTS_PROPERTY_MEMDEF',0)} See related topics

1-2-3: GetEnumString method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_GETENUMSTRING_METHOD_EXSCRIPT',1)} See example

Returns the string associated with a specified enum tag registered by the application. You can use this method to get the enum string associated with an enum tag held in a property.

Syntax

constantname = *application*.**GetEnumString**(*enumtag*)

Parameters

enumtag

Variant. The tag that you use to specify a 1-2-3 enumeration value. This is the tag that the application registers with LotusScript for the given enum constant. For example, the tag $Medium for the EdgeLineWidth property.

Return values

String. The name of the specified constant (for example, "$Medium").

```
' Example: GetEnumString method
' Get the enum constant for an enum tag.

Dim calcModeStr As String
' Set the .CalcMode tag to $Auto.
CurrentDocument.CalcMode = $Auto

' Later, get the string for the current .CalcMode setting.
calcModeStr = CurrentApplication.GetEnumString(CurrentDocument.CalcMode)
' The string calcModeStr now contains "$Auto".
```

1-2-3: GetFieldAlias method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_FIELDAGGREGATETYPE_METHOD_EXSCRIPT',1)} See example

Gets the alias of the given field. The alias is the name shown in the first row of the field in a query table.

Syntax

alias **=** *dataquery*.**GetFieldAlias**(*fieldname*)

Parameters

fieldname

String. The name of the field for which the alias is to be returned.

Return values

String. The alias for the field.

Usage

You can use the field alias to identify the contents of the field.

{button ,AL(`H_123_FIELDALIAS_METHOD_MEMDEF',0)} See related topics

1-2-3: GetKey method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_GETKEY_METHOD_EXSCRIPT',1)} See example

Returns the next key pressed by the user in 1-2-3.

Syntax

key **=** *application***.GetKey**

Parameters

None

Return values

String. A single-character string containing the key, for printable characters entered by the user. A descriptive longer string for function keys, etc. (for example, A returns "A", F6 returns "F6", and CTRL-K returns "Ctrl K").

```
' Example: GetKey method
' Get the next key pressed by the user.
Dim key As String
MessageBox "Click OK and press X key.", MB_OK + MB_ICONINFORMATION, "Script input"
key = CurrentApplication.GetKey
```

1-2-3: GetMenuPosition method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_GETMENU_METHOD_EXSCRIPT',1)} See example

Gets the position of a specified menu within the application's menu bar.

Syntax

position = *application*.**GetMenuPosition**(*menuid*)

Parameters

menuid

Variant (TopLevelMenu enumeration). The ID of the menu whose position is to be returned. The following table lists the allowed values for this parameter.

Value	Description
$FileMenu	The File menu
$EditMenu	The Edit menu
$ViewMenu	The View menu
$CreateMenu	The Create menu
$WindowMenu	The Window menu
$HelpMenu	The Help menu

Return values

Long. A positive integer indicating the menu's position in the menu bar. The value 1 indicates the first position. The value –1 means the menu was not found.

Usage

Use this method to allow for the menu bar changes that 1-2-3 makes in various user contexts. For example, you can't add an item or menu between any special menus that 1-2-3 displays in specific contexts, such as the Range or Sheet menus.

{button ,AL(`H_123_ADDMENU_METHOD_MEMDEF;H_123_GETMENU_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF',0)} See related topics

1-2-3: GetMenu method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

{button ,AL(`H_123_GETMENU_METHOD_EXSCRIPT',1)} See example

Gets the submenu that is in the specified position on the menu.

Syntax

menu = *object*.**GetMenu**(*position*)

Parameters

position

Long. The submenu's position in the menu.

Value	Description
A positive integer	The submenu's position in the menu, counting forward from the beginning. The value 1 means the first position. If *position* exceeds the last menu position, the last menu is returned.
A negative integer	The submenu's position in the menu, counting backward from the end. The value –1 means the last position. If *position* indicates a menu before the first menu, the first menu is returned.

Return values

Menu. The submenu to get.

Usage

If there is any uncertainty about whether the menu position has a submenu or an item, use the GetItemType method to determine this before calling the GetMenu method. The GetMenu method works only when the position has a Menu object. If the position has an item that is not a submenu, you can find the item name with the GetItemText method.

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_REPLACEMENU_METHOD_MEMDEF;H_123_GETITEMTEXT_METHOD_MEMDEF;H_123_GETITEMTYPE_METHOD_MEMDEF;H_123_GETMENUPOSITION_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF',0)} See related topics

```
' Example: GetMenu method, GetMenuPosition, GetItemText, and DisableItem methods
' Get the positions of the Window menu and the New Window menu item, and
' disable the New Window item.
Dim menu1 As Menu                   ' Window menu object
Dim position As Long                ' Position of Window menu in menu bar
' Get the Window menu object.
position = CurrentApplication.GetMenuPosition($WindowMenu)
Set menu1 = CurrentApplication.CurrentMenuBar.GetMenu(position)
' Verify the position of the New Window item and disable it.
If menu1.GetItemText(1) = "&New Window" Then
   menu1.DisableItem 1
Else
   ' The New Window item has been displaced to a different position.
   ' Loop over the other items to find the New Window item and then disable it ...
End If
```

1-2-3: GetRGB method

{button ,AL(`H_123_COLOR_CLASS',0)} See list of classes

{button ,AL(`H_123_GETRGB_METHOD_EXSCRIPT',1)} See example

Returns the RGB value of a Color object.

Syntax

rgbvalue **=** *color*.**GetRGB**

Parameters

None

Return values

Long. The RGB value of the Color object. The following table lists the bit-field values inside this return value.

Value	Description
Bits 0 – 7	Blue
Bits 8 – 15	Green
Bits 16 – 23	Red
Bits 24 – 31	All zero

{button ,AL(`H_123_COLOR_CLASS;H_123_COLORS_PROPERTY_MEMDEF;H_123_RGB_PROPERTY_MEMDEF;H_123_COLORFROMRGB_METHOD_MEMDEF',0)} See related topics

```
' Example: GetRGB method, Colors Property
' Set a Color object to "blue" and get its RGB value.
Dim blueRGB As Long
Dim blueColor As Color
Set blueColor = CurrentApplication.Colors("blue")
blueRGB = blueColor.GetRGB
' You can use this RGB value, for example, to set up an RGB value
' in a different bit-field format for another application.
```

1-2-3: Goto method

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RANGEBORDER_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_GOTO_METHOD_EXSCRIPT',1)} See example

Shows the object. This method does not change the current selection. It moves the cell pointer to the given object, such as a specified cell, a named range, another sheet, another open file, a graphic object, a chart, or a query table.

When you go to another sheet or file, 1-2-3 moves the cell pointer to the cell you last highlighted in that sheet or file.

Syntax

object.**Goto**

Parameters

None

Return values

None

Usage

You can use the TurnTo method to display a different sheet and change the current selection to cell A1 in the new sheet.

{button ,AL(`H_123_ACTIVATE_METHOD_MEMDEF;H_123_SCROLLTOACTIVECELL_METHOD_MEMDEF;H_123_ACTIVE_PROPERTY_MEMDEF;H_123_ACTIVEDOCUMENT_PROPERTY_MEMDEF;H_123_ACTIVEDOCWINDOW_PROPERTY_MEMDEF;H_123_TURNTO_METHOD_MEMDEF',0)} See related topics

```
' Example: Goto, NewButton, NewObject, NewSheet, TurnTo, and Verb methods
' Create several targets in different sheets and run the Goto method on them.

' Create a new document.
CurrentApplication.NewDocument

' Add 3 sheets to the document.
CurrentDocument.NewSheet $After, 3, True

' Create a button in sheet B.
[B].TurnTo
[B].NewButton 375, 840, 1665, 1320
[Button 1].Select

' Create a rectangle in sheet C.
[C].TurnTo
[C].NewRoundedRectangle 420, 1080, 1740, 1650
[Rectangle 1].Select

' Create an OLE object (WordPad) in sheet D.
[D].TurnTo
[D].NewObject 195, 975, 2220, 4425, "WordPad.Document.1",,, False,,,
[OLE 1].Select
Selection.Verb $OLEVerbShow

' Select cell A:A1.
Msgbox "Select cell A:A1."
[A].TurnTo
[A:A1].Select
' Go to the button in sheet B.
Msgbox "Go to the button in sheet B."
[Button 1].Goto
' Go to the rectangle in sheet C.
Msgbox "Go to the rounded rectangle in sheet C."
[Rectangle 1].Goto
' Go to the OLE object in sheet D.
Msgbox "Go to the OLE object in sheet D."
[OLE 1].Goto
' Return to the selected cell in sheet A.
Msgbox "Return to the selected cell in sheet A."
[A].TurnTo
```

1-2-3: GroupSheets method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_GROUPSHEETS_METHOD_EXSCRIPT',1)} See example

Copies the styles and settings from the specified origin sheet to all sheets in the group specified as the start sheet through the end sheet.

Syntax

document.**GroupSheets** *startsheet, endsheet, originsheet*

Parameters

startsheet

Long. The first sheet in the group.

endsheet

Long. The last sheet in the group.

originsheet

Long. The sheet whose styles and settings are to be copied into all the grouped sheets.

Return values

None

Usage

Changes the user makes to styles and settings in one grouped sheet are automatically applied to all sheets in the group.

Sheet numbers in a document start with 0 for sheet [A].

```
' Example: GroupSheets method

' Set a style for the first sheet.
[A].Background.BackColor.ColorName = "ice blue"

' Create 12 sheets after the first one.
CurrentDocument.NewSheet $After, 12, True

' Group sheets 0 through 9 (A through J) together,
' and apply the styles and settings in sheet A to the entire group.
CurrentDocument.GroupSheets 0, 9, 0

' Refresh the window and turn to one of the sheets in the group.
CurrentWindow.Update
[J].TurnTo
```

1-2-3: Group method

{button ,AL(`H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS',0)} See list of classes

{button ,AL(`H_123_GROUP_METHOD_EXSCRIPT',1)} See example

Creates a group object containing the currently selected drawn objects, if there are more than one. It removes the grouped drawn objects from the DrawObjects collection.

Syntax

[].**Group**

Parameters

None

Return values

None

Usage

You can reference the Group object that this method creates by its bracketed name, which 1-2-3 creates in sequence as Group 1, Group 2, and so on.

{button ,AL(`H_123_DRAWOBJECTS_PROPERTY_MEMDEF;H_123_ADDTOSELECTION_METHOD_MEMDEF;H_123_REMOVE_METHOD_MEMDEF;H_123_SELECT_METHOD_MEMDEF;H_123_UNGROUP_METHOD_MEMDEF;',0)} See related topics

```
' Example: Text property; AddPoint, AddToSelection, Group, Move, NewDrawLine,
'          NewEditText, NewPolyline, NewRectangle, Select, and UnGroup methods

' Example 1: Create two graphic objects and then group them.
Dim myRectangle As Rectangle
Dim myEditText As EditText
' Draw a rectangle on sheet A.
Set myRectangle = [A].NewRectangle(1440, 1440, 4800, 3600)
' Draw an edit control on sheet A.
Set myEditText = [A].NewEditText(2160, 2160, 3600, 2880)
myEditText.Text = "Box #1"
' Select the two objects.
myRectangle.Select
myEditText.AddToSelection
' Group the two objects.
[].Group
' The objects stay grouped when you select something else.
[A:B2].Select
' Do something with the group. For example, move it 0.5" in each direction.
[Group 1].Move 720, 720

' Example 2: Use object reference variables.
' After executing the code in Example 1, group another two objects.
Dim myDrawLine As DrawLine
Dim myPolyline As Polyline
Dim dc As DrawCollection
Dim lineGroup As Group
' Create two drawn objects and select them.
Set myDrawLine = [A].NewDrawLine(5760, 1440, 7200, 1440)
Set myPolyline = [A].NewPolyline(5760, 2880, 7200, 2880)
myPolyline.AddPoint 7200, 4340
myDrawLine.Select
myPolyline.AddToSelection
' Group the two objects as a DrawCollection.
Set dc = Selection
dc.group
' Get a Group object reference variable for the new group.
Set lineGroup = Selection
' Select both groups.
[Group 1].Select
dc.AddToSelection
' Do something with the selected groups.
[].Move 1440, 1440
' To work with the individual objects again, delete the group.
lineGroup.UnGroup            ' This method doesn't change the selection.
```

1-2-3: HideColumns method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_UNHIDECOLUMNS_EXSCRIPT',1)} See example

Hides all columns in a range. This method does not change the column.

Syntax

range.**HideColumns**

Parameters

None

Return values

None

Usage

If the sheet that contains the range is part of a group, hiding columns in this sheet hides these columns in all sheets in the group.

If hidden columns are not protected and the sheet or workbook containing them is not locked, commands that enter new data can write over data in the hidden columns.

{button ,AL(`H_123_HIDE_METHOD_MEMDEF;H_123_HIDECOLUMNS_METHOD_MEMDEF;H_123_HIDEROWS_METHOD_MEMDEF;H_123_ISCOLUMNHIDDEN_PROPERTY_MEMDEF;H_123_ISHIDDEN_PROPERTY_MEMDEF;H_123_SHOW_METHOD_MEMDEF',0)} See related topics

1-2-3: HideIconBar method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Hides the specified set of SmartIcons.

Syntax

applicationwindow.**HideIconBar**(*iconsetname*)

Parameters

iconsetname

String. The name of the set of SmartIcons to be hidden.

Return values

None

{button ,AL(`H_123_ISICONBARSHOWING_METHOD_MEMDEF;H_123_SHOWICONBAR_METHOD_MEMDEF',0) } See related topics

1-2-3: HideRows method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_UNHIDECOLUMNS_EXSCRIPT',1)} See example

Hides all rows in the specified range.

Syntax

range.**HideRows**

Parameters

None

Return values

None

Usage

If the sheet that contains the range is part of a group, hiding rows in this sheet hides these rows in all sheets in the group.

If hidden rows are not protected and the sheet or workbook containing them is not locked, commands that enter new data can write over data in the hidden rows.

{button ,AL(`H_123_HIDECOLUMNS_METHOD_MEMDEF;H_123_HIDESHEETS_METHOD_MEMDEF;H_123_ISHIDDEN_PROPERTY_MEMDEF;H_123_SHOW_METHOD_MEMDEF;H_123_SHOWALLSHEETS_METHOD_MEMDEF',0)} See related topics

1-2-3: HideSheet method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWSHEET_METHOD_EXSCRIPT',1)} See example

Hides the current sheet.

Syntax

sheet.**HideSheet**

Parameters

None

Return values

None

Usage

If hidden sheets are not protected and the workbook containing them is not locked, commands that enter new data can write over data in the hidden sheets.

{button ,AL(`H_123_SHOW_METHOD_MEMDEF',0)} See related topics

1-2-3: InsertColumns method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_INSERTCOLUMNS_METHOD_EXSCRIPT',1)} See example

Inserts one or more columns in the current sheet, or inserts a blank range and shifts the selected range right.

Syntax

range.**InsertColumns** ([*inserttype*])

Parameters

inserttype

(Optional) Variant (enumeration). Specifies how 1-2-3 inserts columns. The values are from the InsertColRowType enumeration. The following table lists the allowed values for this parameter.

Value	Description
$Full	Inserts entire columns; default if you omit the parameter.
$Partial	Inserts only the range and moves the existing range to the right.

Return values

None

Usage

1-2-3 inserts new columns to the left of the selected columns or range. Inserted columns have the default column width.

If the current sheet is part of a group, inserting columns in this sheet inserts columns in all sheets in the group.

When you insert columns, 1-2-3 redefines named ranges and, if necessary, adjusts addresses in formulas. If you insert columns into a named range, the named range expands by the number of columns you inserted.

{button ,AL(`H_123_DELETECOLUMNS_METHOD_MEMDEF;H_123_HIDECOLUMNS_METHOD_MEMDEF;H_123_INSERTROWS_METHOD_MEMDEF;H_123_INSERTSHEET_METHOD_MEMDEF',0)} See related topics

```
' Example: InsertColumns and InsertRows methods
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Add some data to cells A:A1 through A:B6 in TestDocument.
    [A:A1].Contents = "InsertRows"
    [A:A2].Contents = "25"
    [A:A3].Contents = "4"
    [A:A4].Contents = "93"
    [A:A5].Contents = "41"
    [A:A6].Contents = "56"
    [A:B1].Contents = "Example"
    [A:B2].Contents = "1025"
    [A:B3].Contents = "104"
    [A:B4].Contents = "1093"
    [A:B5].Contents = "1041"
    [A:B6].Contents = "1056"
'InsertRows $Full
    MessageBox "Insert three full rows across all columns"
    [A1.A3].InsertRows $Full
'InsertRows $Partial
    MessageBox "Insert three partial rows in column A"
    [A1.A3].InsertRows $Partial
'InsertColumns $Full
    MessageBox "Insert a full column"
    [A1].InsertColumns $Full
'InsertColumns $Partial
    MessageBox "Insert a partial column in row 5"
    [A5].InsertColumns $Partial
```

1-2-3: InsertRows method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_INSERTCOLUMNS_METHOD_EXSCRIPT',1)} See example

Inserts one or more rows in the current file, or inserts only the part of the rows covered by the range.

Syntax

range.**InsertRows**(*inserttype*)

Parameters

inserttype

(Optional) Variant (enumeration). Specifies how 1-2-3 inserts rows. The values are from the InsertColRowType enumeration. The following table lists the allowed values for this parameter.

Value	Description
$Full	Inserts entire rows; default if you omit the parameter.
$Partial	Inserts only the range and moves the existing range down.

Return values

None

Usage

1-2-3 inserts new rows above the selected rows or range. Inserted rows have the same height as the row above the inserted rows.

If the current sheet is part of a group, inserting rows in this sheet inserts rows in all sheets in the group.

When you insert rows, 1-2-3 redefines named ranges and, if necessary, adjusts addresses in formulas. If you insert rows into a named range, the named range expands by the number of rows you inserted.

{button ,AL(`H_123_DELETEROWS_METHOD_MEMDEF;H_123_HIDEROWS_METHOD_MEMDEF;H_123_INSERTCOLUMNS_METHOD_MEMDEF',0)} See related topics

1-2-3: IsIconBarShowing method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Tests if a specified set of SmartIcons is visible.

Syntax

boolean = *applicationwindow*.**IsIconBarShowing** (*iconsetname*)

Parameters

iconsetname

String. The name of the set of SmartIcons.

Return value

Variant (boolean). Returns True if *iconsetname* is visible, False if not.

Usage

The ApplicationWindow.IconBarNames property is a collection of names of known icon bars that you can show or hide.

{button ,AL(`H_123_HIDEICONBAR_METHOD_MEMDEF;H_123_SHOWICONBAR_METHOD_MEMDEF;H_123_ICONBARNAMES_PROPERTY_MEMDEF',0)} See related topics

1-2-3: IsSameObject method

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;H_123_PLOT_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_POLYLINE_CLASS;',0)} See list of classes

{button ,AL(`H_123_ISSAMEOBJECT_METHOD_EXSCRIPT',1)} See example

Tests if two objects are the same.

Syntax

boolean = *object*.**IsSameObject**(*otherobject*)

Parameters

otherobject

Variant. An object or expression returning an object that you want to compare to the base object.

Return value

Variant (boolean). Returns True if the objects are the same, False if not.

```
'  Example: IsSameObject method
'  This example creates two color objects and uses
'  the IsSameObject method to test if the two
'  objects are the same.
   ' Declare a variable for the first new color object.
    Dim firstColor As Color
   ' Declare a variable for the second new color object.
    Dim secondColor As Color
   ' Declare a variable to hold the return value of the SameColor method.
    Dim colorissame As Variant
   ' Assign the color blue to the first color object.
    Set firstColor = CurrentApplication.Colors("blue")
   ' Assign the color blue to the second color object.
    Set secondColor = CurrentApplication.Colors("blue")
   ' Compare the color of the first color object with that of the second
   ' color object and return True (because the colors are the same).
    colorissame = firstColor.IsSameObject(secondColor)
    If colorissame = True Then
        Messagebox("Colors are same.")
    Else
        Messagebox("Colors are not the same.")
    End If
```

1-2-3: Item method

{button ,AL(`H_123_CHARTS_CLASS;H_123_COLORS_CLASS;H_123_DATALINKS_CLASS;H_123_DOCUMENTS_CLASS;H_123_DRAWOBJECTS_CLASS;H_123_MAPBINS_CLASS;H_123_MAPS_CLASS;H_123_OLEOBJECTS_CLASS;H_123_PRINTSETTINGSCOLLECTION_CLASS;H_123_RANGES_CLASS;H_123_SHEETS_CLASS;H_123_STRINGS_CLASS;H_123_VERSIONGROUPS_CLASS;H_123_VERSIONS_CLASS;H_123_WINDOWS_CLASS;H_123_BASECOLLECTION_CLASS;H_123_DOCWINDOWS_CLASS;H_123_MAPTEXTENTRIES_CLASS;H_123_QUERYTABLES_CLASS;',0)} See list of classes

{button ,AL(`H_123_ITEM_METHOD_EXSCRIPT;H_123_SHEETS_PROPERTY_EXSCRIPT;H_123_SHORTDAYNAMES_PROPERTY_EXSCRIPT;',1)} See example

Returns the specified member of a collection of elements.

Syntax

indexeditem = *object*.**Item** (*index*)

Parameters

index

Variant. Specifies an index value for the object in the collection.

Return value

indexeditem

Variant. Holds the indexed element.

```
' Example: Item method
' This example uses the Item method to initialize a variable to the item "blue"
' in the Color collection.
    Dim adtcolors As Colors
    Dim curcolor As color
    Set adtcolors = CurrentApplication.Colors
    Set curcolor = adtcolors.Item("blue")
```

1-2-3: Join method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Specifies a join formula to use for a query.

Syntax

query.**Join**([*joinformula*])

Parameters

joinformula

(Optional) String. Specifies a query criteria that compares fields from separate tables. If you omit *joinformula*, this method removes all joined tables from the query.

Return values

None

Usage

In a join formula:

- Precede the field name with the table name and a . (period).
- Enter field names exactly as they appear in the database tables.
- The field names do not have to match, but the two fields must contain the same type of data.
- Entries in one field must match entries in the other field, and one field should not contain duplicate entries.

{button ,AL(`H_123_NEWQUERY_METHOD_MEMDEF;H_123_NEWQUERYTABLE_METHOD_MEMDEF;H_123_QUERYEXTRACT_METHOD_MEMDEF;H_123_QUERYFIND_METHOD_MEMDEF',0)} See related topics

1-2-3: LoadAddin method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_LOADADDIN_METHOD_EXSCRIPT',1)} See example

Reads an add-in into memory.

Syntax

application.**LoadAddin**(*addinname*)

Parameters

addinname

String. Specifies the name of the add-in you want to load.

Return value

None

Usage

The Application.Addins property is a collection of names of the registered add-ins that you can load or unload. Use the IsAddinLoaded method to determine whether an add-in is loaded.

{button ,AL(`H_123_LOADEDADDINS_PROPERTY_MEMDEF;H_123_UNLOADADDIN_METHOD_MEMDEF;H_123_ISADDINLOADED_METHOD_MEMDEF;H_123_ADDINS_PROPERTY_MEMDEF',0)} See related topics

```
' Example: LoadAddin, IsAddinLoaded, UnloadAddin methods
' Load, use, and unload an add-in.
' First, load the add-in.
If CurrentApplication.IsAddinLoaded("C:\Lotus\123\Addins\HTMLbtn.12a") = False Then
   CurrentApplication.LoadAddin "C:\Lotus\123\Addins\HTMLbtn.12a"
End If
' Do something with the add-in ...
' When finished, unload the add-in.
CurrentApplication.UnloadAddin "C:\Lotus\123\Addins\HTMLbtn.12a"
```

1-2-3: MacroRunText method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_MACRORUNTEXT_METHOD_EXSCRIPT',1)} See example

Executes specified text as a 1-2-3 macro.

Syntax

sheet.**MacroRunText**(*macrotext*)

Parameters

macrotext

String. Specifies a 1-2-3 macro.

Return values

None

{button ,AL(`H_123_MACRORUN_METHOD_MEMDEF;H_123_MACROSTEP_PROPERTY_MEMDEF;H_123_MACROTRACE_PROPERTY_MEMDEF',0)} See related topics

```
' Example: MacroRunText method
' This example runs a macro that enters "HELLO" in cell A1
    [].MacroRunText "{SELECT A1}HELLO~"
```

1-2-3: MacroRun method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_MACRORUN_METHOD_EXSCRIPT',1)} See example

Runs the 1-2-3 macro located in the top left cell of the specified range.

Syntax

range.**MacroRun**

Parameters

None

Return values

None

{button ,AL(`H_123_MACRORUNTEXT_METHOD_MEMDEF;H_123_MACROSTEP_PROPERTY_MEMDEF;H_123_MACROTRACE_PROPERTY_MEMDEF',0)} See related topics

```
' Example: MacroRun method
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Enter a macro in A5..A6
    [A:A5].Select
    Selection.Contents = "{SELECT A1}"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "HELLO~"
'Run the macro in A5..A6
    [A:A5..A:A6].MacroRun

'Use the MessageBox statement to display a
'message asking if you want to close the test document.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document
        CurrentDocument.Close False
    End If
```

1-2-3: MakeCurrent method

{button ,AL(`H_123_MAKECURRENT_METHOD_MEMDEF_RT;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS',0)} See list of classes

Makes the specified version or version group the currently displayed version or version group.

Syntax

object.**MakeCurrent**

Parameters

None

Return values

None

{button ,AL(`H_123_VERSION_METHOD_MEMDEF;H_123_VERSIONGROUP_METHOD_MEMDEF;H_123_VERSIONS_METHOD_MEMDEF',0)} See related topics

1-2-3: MatrixInvert method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Creates the inverse of a square matrix.

Syntax

document.**MatrixInvert** *rangetoinvert, outputrange*

Parameters

rangetoinvert

Variant. The range containing the matrix that you want to invert.

outputrange

Variant. The range where you want 1-2-3 to put the results of the matrix inversion.

Caution 1-2-3 writes over any data in *outputrange*.

Return values

None

Usage

The matrix range must have the same number of columns as rows and can contain up to 80 columns and 80 rows.

Matrix inversion algorithms by their nature propagate small errors. Inverting an ill-conditioned matrix (a matrix that contains numbers differing widely in magnitude) may result in large errors. If 1-2-3 cannot invert the matrix at all, LotusScript returns an error.

Inverting 3D matrixes

A 3D matrix includes the same cells in two or more contiguous sheets. When you invert a 3D matrix, the matrix you want to invert and the results range must be square on each sheet, and contain the same number of sheets.

1-2-3 inverts the matrix in each sheet of the 3D range and enters the results in each sheet. The matrixes must be in the same workbook but can be in different sheets. Also, the 3D results matrix must be in the same workbook as the matrixes you're inverting, but can be in different sheets.

For example, 1-2-3 inverts the matrix in the first sheet of the range and enters the results in the first sheet, inverts the matrix in the second sheet and enters the results in the second sheet, and so on.

1-2-3: MatrixMultiply method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Multiplies the values in two input matrixes to create an output matrix that contains the results.

Syntax

document.**MatrixMultiply** *inputmatrix1, inputmatrix2, outputrange*

Parameters

inputmatrix1

Range. The first matrix you want to multiply.

inputmatrix2

Range. The second matrix you want to multiply.

outputrange

Range. The range where you want 1-2-3 to put the result matrix.

Caution 1-2-3 writes over any data in *outputrange*.

Return value

None

Usage

The number of rows in *inputmatrix1* must equal the number of columns in *inputmatrix2* and *outputrange*.

Matrix multiplication algorithms by their nature propagate small errors. Multiplying an ill-converted matrix (a matrix that contains numbers differing greatly in magnitude) or a very large matrix may result in less accurate results.

Multiplying 3D matrixes

A 3D matrix includes the same cells in two or more contiguous sheets. When you multiply 3D matrixes, both matrixes must be 3D, and both must contain the same number of sheets. The matrixes can be in different sheets, or in different workbooks. The 3D results matrix can also be in different sheets or workbooks from the the matrixes you're multiplying.

1-2-3 multiplies the first and second matrixes in each sheet and enters the results in the resulting matrix range in that sheet. For example, 1-2-3 multiplies the range in the first sheet of the first matrix by the range in the first sheet of the second matrix and enters the results in the first sheet, multiplies the range in the second sheet of the first matrix by the range in the second sheet of the second matrix and enters the results in the second sheet, and so on.

{button ,AL(`H_123_MATRIXINVERT_METHOD_MEMDEF',0)} See related topics

1-2-3: Maximize method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_MAXIMIZE_METHOD_EXSCRIPT',1)} See example

Maximizes the window.

Syntax

object.**Maximize**

Parameters

None

Return Values

None

{button ,AL(`H_123_MINIMIZE_METHOD_MEMDEF;H_123_RESTORE_METHOD_MEMDEF',0)} See related topics

```
'Maximize method
'Maximizes the 1-2-3 application window
CurrentApplication.ApplicationWindow.Maximize
```

1-2-3: MergeVersions method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Copies versions from named ranges in the source workbook to ranges of the same size and with the same names in the destination workbook.

Syntax

document.**MergeVersions** ([*sourcefile*], [*datefilter*], [*userfilter*], [*tablelocation*])

Parameters

sourcefile

(Optional) String. The name of the file that contains the versions and version groups you want to merge. The default is the current file.

datefilter

(Optional) Variant. Tells 1-2-3 to merge only versions and version groups created or modified on or after a particular date. The default is to merge all version and version groups.

userfilter

(Optional) String. Tells 1-2-3 to merge only versions and version groups created or last modified by a particular user. The default is to merge all version and version groups.

tablelocation

(Optional) Variant. The range where you want 1-2-3 to create the table of merge results. If you omit *tablelocation*, 1-2-3 does not create a table of merge results.

Caution *tablelocation* occupies one column and as many rows as there are merge results, plus one blank row. 1-2-3 writes over any existing data in *tablelocation*.

Return value

None

Usage

1-2-3 does not merge hidden versions and version groups in a locked file.

If a version or version group in the active file has the same name, creation date, last modified date, and last user as a version or version group in *sourcefile*, the version or scenario in the *sourcefile* is not merged.

If *tablelocation* calls for more rows than there are rows remaining in the sheet, 1-2-3 truncates the table.

1-2-3: Minimize method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_MINIMIZE_METHOD_EXSCRIPT',1)} See example

Minimizes the window.

Syntax

object.**Minimize**

Parameters

None

Return values

None

{button ,AL(`H_123_MAXIMIZE_METHOD_MEMDEF;H_123_RESTORE_METHOD_MEMDEF',0)} See related topics

```
'Minimize method
'This function lays the groundwork for an application that runs
'in the background. After opening a document, you can minimize 1-2-3.

'Minimize 1-2-3
CurrentApplication.ApplicationWindow.Minimize
```

1-2-3: ModifyNamedStyle method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_MODIFYNAMEDSTYLE_METHOD_EXSCRIPT',1)} See example

Modifies the style of an existing named style by redefining the attributes of the named style to match the attributes of the upper left cell of the range.

Syntax

range.**ModifyNamedStyle** (*stylename*)

Parameters

stylename

String. Name of the style you want to modify.

Return value

None

{button ,AL(`H_123_SETSTYLESOURCE_METHOD_MEMDEF;H_123_STYLE_PROPERTY_MEMDEF;H_123_STYLENAME_PROPERTY_MEMDEF;H_123_STYLERANGE_PROPERTY_MEMDEF;H_123_STYLESOURCE_PROPERTY_MEMDEF',0)} See related topics

```
' Example: ModifyNamedStyle method
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Create two labels (one for each named style):
' Style1 and Style2.
    MessageBox "Create labels for styles Style1 and Style2."
    [A:A1].Select
    Selection.Contents = "Style1"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "Style2"

' Make each style distinctive.
    MessageBox "Make each style distinctive."
    [A:A1].Select
    Selection.Font.FontColor.ColorName = "blue"
    Selection.Font.Bold = True
    Selection.Background.BackColor.ColorName = "ice blue"
    Selection.DefineNamedStyle "Style1"
    Selection.StyleName = "Style1"
    [A:A2].Select
    Selection.Font.Italic = True
    Selection.Font.FontColor.ColorName = "red"
    Selection.Background.BackColor.ColorName = "blush"
    Selection.DefineNamedStyle "Style2"
    Selection.StyleName = "Style2"

' Select C1 and give it styles
    MessageBox "Select C1 and give it styles."
    [A:C1].Select
    Selection.Font.DoubleUnderline = True
    Selection.Font.Size = 14
    Selection.Font.FontColor.ColorName = "dark green"
    Selection.Background.BackColor.ColorName = "pale green"
'Modify Style1 to give it the styles in C1
    MessageBox "Modify Style1 to give it the styles in C1."
    [C1].ModifyNamedStyle "Style1"

'Use the MessageBox statement to display a
'message asking if you want to close the test document.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
```

```
If answer% = 6 Then
'If the answer is 6 (IDYES), close the test document
    CurrentDocument.Close False
End If
```

1-2-3: MoveCellPointer method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_MOVECELLPOINTER_METHOD_EXSCRIPT',1)} See example

Moves the cell pointer.

Syntax

sheet.**MoveCellPointer** *direction, repeatcount*

Parameters

direction

Variant (SheetDirection enumeration). Specifies the direction and type of movement. The following table lists the allowed arguments for this parameter.

Value	Description
$Left	Moves the cell pointer left one column.
$Right	Moves the cell pointer right one column.
$Up	Moves the cell pointer up one row.
$Down	Moves the cell pointer down one row.
$Home	Moves the cell pointer to cell A1 in the current sheet.
$FirstCell	Moves the cell pointer to cell A:A1 in the current file.
$PgLeft	Moves the cell pointer left the number of columns currently visible in the window.
$PgRight	Moves the cell pointer right the number of columns currently visible in the window.
$PgUp	Moves the cell pointer up the number of rows currently visible in the window.
$PgDown	Moves the cell pointer down the number of rows currently visible in the window.
$EndUp	Moves the cell pointer up to a cell that contains data and is next to a blank cell.
$EndRight	Moves the cell pointer right to a cell that contains data and is next to a blank cell.
$EndLeft	Moves the cell pointer left to a cell that contains data and is next to a blank cell.
$EndDown	Moves the cell pointer down to a cell that contains data and is next to a blank cell.
$EndHome	Moves the cell pointer to the bottom right corner of the sheet's active area.
$LastCell	Moves the cell pointer to the

	bottom right corner of the last sheet's active area.
$EndForward	Moves the cell pointer to the cell you last highlighted in the first active file.
$EndBackward	Moves the cell pointer to the cell you last highlighted in the last active file.
$Forward	Moves the cell pointer to the cell you last highlighted in the next active file.
$Backward	Moves the cell pointer to the cell you last highlighted in the previous active file.

repeatcount

Long. The number of times to repeat the movement.

Return value

None

{button ,AL(`H_123_SCROLLTOACTIVECELL_METHOD_MEMDEF',0)} See related topics

```
' Example: MoveCellPointer method
    .MoveCellPointer $Right,5
    .MoveCellPointer $Down,5
    .MoveCellPointer $Left,5
    .MoveCellPointer $Up,5
```

1-2-3: MoveOrigin method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_MOVEORIGIN_METHOD_EXSCRIPT',1)} See example

Scrolls to display a different cell in the top left corner of the sheet. When you use MoveOrigin, a different location in the sheet is displayed, but the cell pointer does not move.

Syntax

sheet.**MoveOrigin** (*direction,* [*repeatcount*])

Parameters

direction

Variant (enumeration). Specifies the direction and type of movement. The values are from the enumeration SheetDirection. The following table lists the allowed values for this parameter.

Value	Description
$Left	Moves the cell pointer left one column.
$Right	Moves the cell pointer right one column.
$Up	Moves the cell pointer up one row.
$Down	Moves the cell pointer down one row.
$Home	Moves the cell pointer to cell A1 in the current sheet.
$FirstCell	Moves the cell pointer to cell A:A1 in the current file.
$PgLeft	Moves the cell pointer left the number of columns currently visible in the window.
$PgRight	Moves the cell pointer right the number of columns currently visible in the window.
$PgUp	Moves the cell pointer up the number of rows currently visible in the window.
$PgDown	Moves the cell pointer down the number of rows currently visible in the window.
$EndUp	Moves the cell pointer up to a cell that contains data and is next to a blank cell.
$EndRight	Moves the cell pointer right to a cell that contains data and is next to a blank cell.
$EndLeft	Moves the cell pointer left to a cell that contains data and is next to a blank cell.
$EndDown	Moves the cell pointer down to a cell that contains data and is next to a blank cell.
$EndHome	Moves the cell pointer to the bottom right corner of the sheet's active area.

$LastCell	Moves the cell pointer to the bottom right corner of the last sheet's active area.
$EndForward	Moves the cell pointer to the cell you last highlighted in the first active file.
$EndBackward	Moves the cell pointer to the cell you last highlighted in the last active file.
$Forward	Moves the cell pointer to the cell you last highlighted in the next active file.
$Backward	Moves the cell pointer to the cell you last highlighted in the previous active file.

repeatcount

(Optional) . The number of times to repeat the movement.

Return values

None

{button ,AL(`H_123_MOVE_METHOD_MEMDEF;H_123_MOVECELLPOINTER_METHOD_MEMDEF;H_123_MOVEPOINT_METHOD_MEMDEF',0)} See related topics

```
' Example: MoveOrigin method
    .MoveOrigin $Right,5
    .MoveOrigin $Left,5
```

1-2-3: Move method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_PLOT_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RECTANGLE_CLASS;',0)} See list of classes

{button ,AL(`H_123_MOVE_METHOD_EXSCRIPT;',1)} See example

For graphic objects, moves an object by a given distance. For an ApplicationWindow, DocWindow, or Window object, moves the top left corner of the specified window to a new coordinate.

Syntax

For window objects: *window*.**Move** *left*, *top*

For graphic objects: *object*.**Move** *horizontal, vertical*

Parameters

For graphic objects:

horizontal

Long. The number of twips to move the object left or right. Specify a positive value to move the object to the right; specify a negative value to move the object to the left.

vertical

Long. The number of twips to move the object up or down. Specify a positive value to move the object down; specify a negative value to move the object up.

For window objects:

left

Long. The horizontal coordinate (x-axis) for the left edge of the window.

top

Long. The vertical coordinate (y-axis) for the top edge of the window.

Return value

None

Usage

If the window is currently maximized, the move occurs after the window is restored to its original size.

```
'  Example: Move method (windows)
    CurrentApplication.ApplicationWindow.Move 100,300
'  Example: Move method (graphic object)
    MessageBox("Create a button.")
    [A].NewButton 400,1000,1290,1425
    MessageBox("Move the button.")
    [Button 1].Move 300,1200
    MessageBox("Delete the button.")
    [Button 1].Clear
```

1-2-3: NewApproachConnection method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWAPPROACHCONNECTION_METHOD_EXSCRIPT',1)} See example

Creates a new ApproachConnection object.

Syntax

Set *approachconnection* **=** *sheet*.**NewApproachConnection**(*left, top, right, bottom, objecttype,* [*filename*], [*link*], [*displayasicon*]*,* [*iconfilename*]*,* [*iconindex*]*,* [*iconlabel*]*, inputrange*)

Parameters

left

Long. The left position of the object's bounding rectangle, in units of twips. If you omit both *left* and *top*, the ApproachConnection object is created at the position of the current selection.

top

Long. The top position of the object's bounding rectangle, in units of twips. If you omit both *left* and *top*, the ApproachConnection object is created at the position of the current selection.

right

Long. The right position of the object's bounding rectangle, in units of twips. If you omit both *right* and *bottom*, the ApproachConnection object is created with a default size.

bottom

Long. The bottom position of the object's bounding rectangle, in units of twips. If you omit both *right* and *bottom*, the ApproachConnection object is created with a default size.

objecttype

String. The object type as it appears in the Create - Object dialog box. You can specify the *objecttype* or the *filename* argument, but not both.

filename

(Optional, not applicable for this method) String. The name of the file to embed or link. You can specify the *objecttype* or the *filename* argument, but not both.

link

(Optional, not applicable for this method) Variant (Boolean). Specify True to link the file, False to embed the file specified by the *filename* argument. The default is False.

displayasicon

(Optional) Variant (Boolean). Specify True to display the object as an icon, False to display the object's content, for forms and crosstabs. The default is False.

iconfilename

(Optional, not applicable for this method) String. The name of an executable file containing the icon to be displayed for the object. Only relevant if *displayasicon* is True.

iconindex

(Optional, not applicable for this method) Integer. The index that specifies the icon within the file specified by *iconfilename*. Only relevant if *displayasicon* is True.

iconlabel

(Optional, not applicable for this method) String. Text to display below the icon. Only relevant if *displayasicon* is True.

inputrange

Variant. The range containing the source data for the Approach object.

Return values

ApproachConnection. A new ApproachConnection object.

{button ,AL(`H_123_NEWOBJECT_METHOD_MEMDEF;H_123_NEWQUERY_METHOD_MEMDEF;H_123_NEWQUERYTABLE_METHOD_MEMDEF',0)} See related topics

1-2-3: NewArc method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWARC_METHOD_EXSCRIPT',1)} See example

Creates a new Arc object.

Syntax

Set *arc* **=** *sheet***.NewArc**(*left, top, right, bottom*)

Parameters

left

Long. The left position of the arc's bounding rectangle in units of twips.

top

Long. The top position of the arc's bounding rectangle in units of twips.

right

Long. The right position of the arc's bounding rectangle in units of twips.

bottom

Long. The bottom position of the arc's bounding rectangle in units of twips.

Return values

Arc. A new instance of the Arc class.

1-2-3: NewArrow method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_FLIPLEFTRIGHT_METHOD_EXSCRIPT',1)} See example

Creates a new DrawLine object with an arrowhead on its second endpoint.

Syntax

Set *drawline* **=** *sheet***.NewArrow**(*xposition1, yposition1, xposition2, yposition2*)

Parameters

xposition1

Long. The horizontal position of the arrow's first endpoint, in units of twips.

yposition1

Long. The vertical position of the arrow's first endpoint, in units of twips.

xposition2

Long. The horizontal position of the arrow's second endpoint, in units of twips.

yposition2

Long. The vertical position of the arrow's second endpoint, in units of twips.

Return values

DrawLine. A new instance of the DrawLine class, with its Arrow property set to $Head.

{button ,AL(`H_123_NEWDRAWLINE_METHOD_MEMDEF;H_123_ARROW_PROPERTY_MEMDEF',0)} See related topics

1-2-3: NewButton method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Creates a new ButtonControl object.

Syntax

Set *buttoncontrol* **=** *sheet*.**NewButton**(*left, top, right, bottom*)

Parameters

left

Long. The left position of the button's bounding rectangle, in units of twips.

top

Long. The top position of the button's bounding rectangle, in units of twips.

right

Long. The right position of the button's bounding rectangle, in units of twips.

bottom

Long. The bottom position of the button's bounding rectangle, in units of twips.

Return values

ButtonControl. A new instance of the ButtonControl class.

1-2-3: NewChart method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWDRAWLAYER_PROPERTY_EXSCRIPT;H_123_NEWCHART_METHOD_EXSCRIPT',1)} See example

Creates a new Chart object.

Syntax

Set *chart* **=** *sheet*.**NewChart**(*left, top, right, bottom*, *inputrange*)

Parameters

left

Long. The left position of the chart's bounding rectangle, in units of twips.

top

Long. The top position of the chart's bounding rectangle, in units of twips.

right

Long. The right position of the chart's bounding rectangle, in units of twips.

bottom

Long. The bottom position of the chart's bounding rectangle, in units of twips.

inputrange

Variant. The input range of data to be charted.

Return values

Chart. A new instance of the Chart class.

1-2-3: NewDataLink method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWDATALINK_METHOD_EXSCRIPT',1)} See example

Creates a new DataLink object, linked to an OLE server.

Syntax

Set *datalink* **=** *document*.**NewDataLink**([*linkname*]*, filename, itemname,* [*format*]*,* [*autoupdate*])

Parameters

linkname

(Optional) String. Name for the link.

filename

String. The name of the external file to link. The file application must be an OLE server.

itemname

String. The name of the source item in the link file. The source item must be in text, WK1, or WK3 format.

format

(Optional) Variant (ClipboardFormat enumeration). The Clipboard data format to use. The link returns text or cell data in one of the following allowed formats.

Value	Description
$TextFormat	Text
$WK1Format	Lotus 1-2-3 Release 2 file
$WK3Format	Lotus 1-2-3 file; Release 1 for Windows, Releases 3, 4 for DOS

autoupdate

(Optional) Variant (Boolean). Specify True for automatic link updates, False for manual link updates.

Return value

DataLink. A new instance of the DataLink class.

{button ,AL(`H_123_NEWOBJECT_METHOD_MEMDEF;H_123_NEWAPPROACHCONNECTION_METHOD_MEMDEF;H_123_NEWQUERY_METHOD_MEMDEF;H_123_NEWQUERYTABLE_METHOD_MEMDEF',0)} See related topics

1-2-3: NewDocument method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_ACTIVATE_METHOD_EXSCRIPT',1)} See example

Creates a new Document object (workbook) in this application and displays it. The new document is not associated with a preexisting file.

Syntax

Set *document* **=** *application*.**NewDocument**([*name*], [*location*], [*smartmaster*], [*smartlocation*], [*smartpassword*], [*adjacentdocument*], [*openmode*])

Parameters

name

(Optional) String. A name for the document. If you don't specify a document *name*, "Untitled", "Untitled1", "Untitled2", and so on are used in sequence.

location

(Optional, not currently used) Variant. The location (path) for the new document if and when it is saved. 1-2-3 only accepts a string for this parameter.

smartmaster

(Optional) String. A SmartMaster template on which to base the new document.

smartlocation

(Optional) String. The directory path for the SmartMaster template.

smartpassword

(Optional) String. A potential password for the SmartMaster used.

adjacentdocument

(Optional) Document. A document in memory before or after which to open the new document.

openmode

(Optional) Variant (OpenMode enumeration). Specifies whether to open the new document before or after the document specified by *adjacentdocument*. The default is $OpenModeAfter. The following table lists the allowed values for this parameter.

Value	Description
$OpenModeBefore	Open the new document before the document specified by *adjacentdocument*.
$OpenModeAfter	Open the new document after the document specified by *adjacentdocument*.

Return values

Document. A new instance of the Document class.

Usage

No permanent file for the document is created on disk until you call the SaveAs method with the *name* argument.

1-2-3 creates a new Document object for every workbook at the time the workbook is opened by the user or by script. Running the NewDocument method is the way to create a new workbook by script. Running the OpenDocument method is the way to open an existing workbook file by script.

1-2-3: NewDocWindow method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Creates a new workbook window with the same view of the current sheet as the current workbook window.

Syntax

Set *docwindow* **=** *document*.**NewDocWindow**

Parameters

None

Return values

DocWindow. A new instance of the DocWindow class.

{button ,AL(`H_123_NEWDOCUMENT_METHOD_MEMDEF',0)} See related topics

1-2-3: NewDrawLine method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_GROUP_METHOD_EXSCRIPT',1)} See example

Creates a new DrawLine object.

Syntax

Set *drawline* **=** *sheet***.NewDrawLine**(*xposition1, yposition1, xposition2, yposition2*)

Parameters

xposition1

Long. The horizontal position of the drawn line's first endpoint, in units of twips.

yposition1

Long. The vertical position of the drawn line's first endpoint, in units of twips.

xposition2

Long. The horizontal position of the drawn line's second endpoint, in units of twips.

yposition2

Long. The vertical position of the drawn line's second endpoint, in units of twips.

Return value

DrawLine. A new instance of the DrawLine class.

1-2-3: NewEditText method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_GROUP_METHOD_EXSCRIPT',1)} See example

Creates a new EditText object.

Syntax

Set *edittext* **=** *sheet***.NewEditText**(*left, top, right, bottom*)

Parameters

left

Long. The left position of the text box's bounding rectangle, in units of twips.

top

Long. The top position of the text box's bounding rectangle, in units of twips.

right

Long. The right position of the text box's bounding rectangle, in units of twips.

bottom

Long. The bottom position of the text box's bounding rectangle, in units of twips.

Return values

EditText. A new instance of the EditText class.

{button ,AL(`H_123_EDITTEXT_CLASS',0)} See related topics

1-2-3: NewEllipse method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_BOUNDS_METHOD_EXSCRIPT',1)} See example

Creates a new Ellipse object.

Syntax

Set *ellipse* **=** *sheet*.**NewEllipse**(*left, top, right, bottom*)

Parameters

left

Long. The left position of the ellipse's bounding rectangle, in units of twips.

top

Long.The top position of the ellipse's bounding rectangle, in units of twips.

right

Long. The right position of the ellipse's bounding rectangle, in units of twips.

bottom

Long. The bottom position of the ellipse's bounding rectangle, in units of twips.

Return values

Ellipse. A new instance of the Ellipse class.

{button ,AL(`H_123_ELLIPSE_CLASS',0)} See related topics

1-2-3: NewFreehand method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Creates a new Freehand object and draws a line connecting its first two points.

Syntax

Set *freehand* **=** *sheet*.**NewFreehand**(*xposition1, yposition1, xposition2, yposition2*)

Parameters

xposition1

Long. The horizontal position of the freehand drawing's first point, in units of twips.

yposition1

Long. The vertical position of the freehand drawing's first point, in units of twips.

xposition2

Long. The horizontal position of the freehand drawing's second point, in units of twips.

yposition2

Long. The vertical position of the freehand drawing's second point, in units of twips.

Return values

Freehand. A new instance of the Freehand class.

Usage

This method creates a default name "Freehand *n*" for the new Freehand object, where *n* is the new number of Freehand objects in the sheet. You can refer to the new Freehand object as [Freehand *n*].

1-2-3: NewMap method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWMAP_METHOD_EXSCRIPT',1)} See example

Creates a new Map object.

Syntax

Set *map* **=** *sheet***.NewMap**(*left, top, right, bottom,* [*datarange*], [*mapname*])

Parameters

left

Long. The left position of the map's bounding rectangle, in units of twips.

top

Long. The top position of the map's bounding rectangle, in units of twips.

right

Long. The right position of the map's bounding rectangle, in units of twips.

bottom

Long. The bottom position of the map's bounding rectangle, in units of twips.

datarange

(Optional) Variant. Range of data to be mapped. If you don't supply this argument, the current range selection is used.

mapname

(Optional) String. Name of the new map. This must be the name of an installed map (for example, "World Countries"). If you don't supply this argument, and the data range doesn't clearly indicate a specific map, the Map Types dialog box prompts the user for the name.

Return values

Map. A new instance of the Map class.

Usage

You can determine the names of the installed maps by looking at the Map Types dialog box displayed for an empty range selection, or by consulting the Windows registry.

1-2-3: NewMenu method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDMENU_METHOD_EXSCRIPT',1)} See example

Creates an empty menu.

Syntax

Set *menu* **=** *application***.NewMenu**

Parameters

None

Return values

Menu. A new instance of the Menu class.

Usage

You can populate the menu with non-submenu items using the AddItem method, and with submenus using the AddMenu method.

1-2-3: NewMenuBar method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Creates an empty menu bar.

Syntax

Set *menubar* **=** *application*.**NewMenuBar**

Parameters

None

Return values

MenuBar. A new instance of the MenuBar class.

1-2-3: NewNamedPrintSettings method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETENAMEDPRINTSETTINGS_METHOD_EXSCRIPT',1)} See example

Creates a new named PrintSettings object and initializes it. Adds the new object to the document's NamedPrintSettings collection.

Syntax

Set *printsettings* **=** *document***.NewNamedPrintSettings**(*settingsname,* [*saveprintselection*], [*sourcesettings*])

Parameters

settingsname

String. The name of the new PrintSettings object. If the print settings name you specify already exists, 1-2-3 generates an error.

saveprintselection

(Optional) Variant (Boolean). Specify True to save the print selection as part of the style, False not to save it.

sourcesettings

(Optional) PrintSettings. A PrintSettings object to use as the source of print settings for initializing values in the new object. If you don't specify an object, the new object is initialized with values from the CurrentPrintSettings property of the document.

Return values

PrintSettings. A new instance of the PrintSettings class.

Usage

A named PrintSettings object holds a named print style.

{button ,AL(`H_123_DELETENAMEDPRINTSETTINGS_METHOD_MEMDEF',0)} See related topics

1-2-3: NewObject method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWOBJECT_METHOD_EXSCRIPT',1)} See example

Creates a new OLE object, either linked or embedded.

Syntax

Set *oleobject* **=** *sheet*.**NewObject**([*left*], [*top*], [*right*], [*bottom*], *objecttype, filename,* [*link*], [*displayasicon*], [*iconfilename*], [*iconindex*], [*iconlabel*])

Parameters

left

(Optional) Long. The left position of bounding rectangle, in units of twips. If you omit this argument, the object is created at the leftmost position.

top

(Optional) Long. The top position of bounding rectangle, in units of twips. If you omit this argument, the object is created at the topmost position.

right

(Optional) Long. The right position of bounding rectangle, in units of twips. If you omit both *right* and *bottom*, the object is created with a default size.

bottom

(Optional) Long. The bottom position of bounding rectangle, in units of twips. If you omit both *right* and *bottom*, the object is created with a default size.

objecttype

String. The object type, or class definition, registered with the operating system for the server application. For example, "Paint.Picture" for a .BMP file. You must specify either the *objecttype* or the *filename* argument, but not both.

filename

String. The name of the file to embed or link, without path information. You must specify either the *objecttype* or the *filename* argument, but not both. 1-2-3 looks for this file in the directory containing the 1-2-3 application.

link

(Optional) Variant (Boolean). Specify True to link the file specified by the *filename* argument, False to embed the file. The default is False. This argument only has meaning if you specify the *filename* argument.

displayasicon

(Optional) Variant (Boolean). Specify True to display the object as an icon, False to display the object's content. The default is False.

iconfilename

(Optional) String. The name of an executable file containing the icon to be displayed for the object. Only relevant if *displayasicon* is True.

iconindex

(Optional) Integer. The index that specifies the icon within the file specified by *iconfilename*. Only relevant if *displayasicon* is True. The default is zero.

iconlabel

(Optional) String. Text to display below the icon. Only relevant if *displayasicon* is True.

Return values

OLEObject. A new instance of the OLEObject class.

1-2-3: NewPicture method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Creates a new Picture object in the file.

Syntax

Set *picture* **=** *sheet*.**NewPicture**(*filename, filetype, left, top, right, bottom*)

Parameters

filename

String. The path and name of the file containing the picture.

filetype

Variant (DocType enumeration). The format of the picture file. The following table lists the allowed values for this parameter.

Value	Description
$DocType_BMP	Bitmap (Paint or Paintbrush)
$DocType_CGM	ANSI metafile (CGM)
$DocType_PIC	1-2-3 picture
$DocType_WMF	Windows metafile

left

Long. The left position of the picture's bounding rectangle, in units of twips.

top

Long. The top position of the picture's bounding rectangle, in units of twips.

right

Long. The right position of the picture's bounding rectangle, in units of twips.

bottom

Long. The bottom position of the picture's bounding rectangle, in units of twips.

Return values

Picture. A new instance of the Picture class.

1-2-3: NewPolygon method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDPOINT_METHOD_EXSCRIPT',1)} See example

Creates a new Polygon object and draws its first line between the specified points.

Syntax

Set *polygon* **=** *sheet*.**NewPolygon**(*xposition1, yposition1, xposition2, yposition2*)

Parameters

xposition1

Long. Horizontal position of the polygon's first point, in units of twips.

yposition1

Long. Vertical position of the polygon's first point, in units of twips.

xposition2

Long. Horizontal position of the polygon's second point, in units of twips.

yposition2

Long. Vertical position of the polygon's second point, in units of twips.

Return values

Polygon. A new instance of the Polygon class.

Usage

Use the AddPoint method to add more points to the polygon. 1-2-3 closes the polygon automatically, by drawing a line between the first and last points.

{button ,AL(`H_123_NEWDRAWLINE_METHOD_MEMDEF;H_123_NEWPOLYLINE_METHOD_MEMDEF;H_123_NEWRECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: NewPolyline method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Creates a new Polyline object and draws its first line between the specified points.

Syntax

Set *polyline* **=** *sheet*.**NewPolyline**(*xposition1, yposition1, xposition2, yposition2*)

Parameters

xposition1

Long. The horizontal position of the polyline's first point, in units of twips.

yposition1

Long. The vertical position of the polyline's first point, in units of twips.

xposition2

Long. The horizontal position of the polyline's second point, in units of twips.

yposition2

Long. The vertical position of the polyline's second point, in units of twips.

Return values

Polyline. A new instance of the Polyline class.

Usage

Use the AddPoint method to add more points to the polyline.

1-2-3: NewQuery method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CREATECOMPUTEDFIELD_METHOD_EXSCRIPT',1)} See example

Creates a new DataQuery object in the document.

Syntax

Set *query* **=** *document***.NewQuery**([*queryname*], [*sourcetablename*])

Parameters

queryname

(Optional) String. The name of the new query. The name can have up to 15 characters. If you don't specify this argument, 1-2-3 generates a name such as "Query 1".

sourcetablename

(Optional) String. The base source table name or address for the query. If you don't specify this argument, most query commands and properties are invalid until you set the BaseSourceTable property for the query.

Return values

DataQuery. A new instance of the DataQuery class.

1-2-3: NewQueryTable method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWQUERYTABLE_METHOD_EXSCRIPT;H_123_REFRESHOUTPUT_METHOD_EXSCRIPT', 1)} See example

Creates a new QueryTable object.

Syntax

Set *querytable* **=** *sheet*.**NewQueryTable**(*left, top, right, bottom, objecttype,* [*filename*], [*link*], [*displayasicon*], [*iconfilename*], [*iconindex*], [*iconlabel*], [*inputrange*], [*outputrange*])

Parameters

left

Long. The left position of the bounding rectangle, in units of twips. If you omit both *left* and *top*, the QueryTable object is created at the position of the current selection.

top

Long. The top position of the bounding rectangle, in units of twips. If you omit both *left* and *top*, the QueryTable object is created at the position of the current selection.

right

Long. The right position of the bounding rectangle, in units of twips. If you omit both *right* and *bottom*, the QueryTable object is created with a default size.

bottom

Long. The bottom position of the bounding rectangle, in units of twips. If you omit both *right* and *bottom*, the QueryTable object is created with a default size.

objecttype

String. The object type as it appears in the Create - Object dialog box. You can specify the *objecttype* or the *filename* argument, but not both.

filename

(Optional, not applicable to this method) String. The name of a file to embed or link. You can specify the *objecttype* or the *filename* argument, but not both.

link

(Optional, not applicable to this method) Variant (Boolean). Specify True to link the file specified by the *filename* argument, False to embed the file.

displayasicon

(Optional, not applicable to this method) Variant (Boolean). Specify True to display the object as an icon, False to display the object's content.

iconfilename

(Optional, not applicable to this method) String. The name of an executable file containing the icon to be displayed for the object. Only relevant if *displayasicon* is True.

iconindex

(Optional, not applicable to this method) Integer. The index that specifies the icon within the file specified by *iconfilename*. Only relevant if *displayasicon* is True.

iconlabel

(Optional, not applicable to this method) String. Text to display below the icon. Only relevant if *displayasicon* is True.

inputrange

(Optional) Variant. The input database table for the query.

outputrange

(Optional) Variant. The output range for query table results.

Return values

QueryTable. A new instance of the QueryTable class.

1-2-3: NewRectangle method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_GROUP_METHOD_EXSCRIPT',1)} See example

Creates a new Rectangle object.

Syntax

Set *rectangle* **=** *sheet*.**NewRectangle**(*left, top, right, bottom*)

Parameters

left

Long. The horizontal position of the rectangle's left side, in units of twips.

top

Long. The vertical position of the rectangle's top side, in units of twips.

right

Long. The horizontal position of the rectangle's right side, in units of twips.

bottom

Long. The vertical position of the rectangle's bottom side, in units of twips.

Return values

Rectangle. A new instance of the Rectangle class.

{button ,AL(`H_123_NEWROUNDRECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: NewRoundedRectangle method

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

Creates a new Rectangle object with rounded corners.

Syntax

Set *rectangle* **=** *sheet*.**NewRoundedRectangle**(*left, top, right, bottom*)

Parameters

left

Long. The horizontal position of the rectangle's left side, in units of twips.

top

Long. The vertical position of the rectangle's top side, in units of twips.

right

Long. The horizontal position of the rectangle's right side, in units of twips.

bottom

Long. The vertical position of the rectangle's bottom side, in units of twips.

Return values

Rectangle. A new instance of the Rectangle class, with the Rounded property set to True.

{button ,AL(`H_123_NEWRECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: NewSheet method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWSHEET_METHOD_EXSCRIPT',1)} See example

Inserts one or more new sheets into a document.

Syntax

Set *sheet* **=** *document***.NewSheet**(*sheetposition, sheetcount, current*)

Parameters

sheetposition

Variant (InsertType enumeration). Specifies where to insert the new sheets. The following table lists the allowed values for this parameter.

Value	Description
$Before	Insert before the current sheet.
$After	Insert after the current sheet.
$First	Insert as the first sheet.
$Last	Insert as the last sheet.

sheetcount

Long. The number of sheets to insert.

current

Variant (Boolean). If True, the first new sheet is made current. If False, the existing current sheet remains current.

Return values

Sheet. The Sheet object for the first new sheet.

1-2-3: NewVersion method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_NEWVERSION_METHOD_EXSCRIPT',1)} See example

Creates a new versioned instance of this range. When you create a version for the first time on a non-versioned range, two versions are actually created: the original, with a default name, and a second version whose name is determined by the *versionname* argument.

Syntax

Set *version* **=** *range***.NewVersion**([*versionname*])

Parameters

versionname

(Optional) String. The name for the new version. If you don't specify this argument, the version is assigned the next system-generated name (for example, "Version 3").

Return values

Version. A new Version object for this range.

{button ,AL(`H_123_DELETEVERSION_METHOD_MEMDEF',0)} See related topics

1-2-3: NewVersionGroup method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_ADDVERSION_METHOD_EXSCRIPT',1)} See example

Creates a new version group within the document.

Syntax

Set *versiongroup* = *document*.**NewVersionGroup**([*versiongroupname*])

Parameters

versiongroupname

(Optional) String. The name of the new version group. If you don't supply this argument, a system default name (in the form "VersionGroup 1") is generated.

Return values

VersionGroup. A new instance of the VersionGroup class.

{button ,AL(`H_123_DELETEVERSION_METHOD_MEMDEF;H_123_MERGEVERSIONS_METHOD_MEMDEF;H_123_NEWVERSIONGROUP_METHOD_MEMDEF;H_123_NEWVERSION_METHOD_MEMDEF;H_123_REMOVEVERSION_METHOD_MEMDEF;H_123_REPORTVERSION_METHOD_MEMDEF;H_123_VERSIONGROUP_METHOD_MEMDEF;H_123_VERSIONS_METHOD_MEMDEF;H_123_VERSION_METHOD_MEMDEF;H_123_CURRENTVERSION_PROPERTY_MEMDEF;H_123_LASTVERSIONGROUP_PROPERTY_MEMDEF;H_123_SHOWVERSIONBORDERS_PROPERTY_MEMDEF;H_123_VERSIONBORDERSVISIBLE_PROPERTY_MEMDEF;',0)} See related topics

1-2-3: Next method

{button ,AL(`H_123_BASECOLLECTION_CLASS;H_123_CHARTS_CLASS;H_123_COLORS_CLASS;H_123_DATALINKS_CLASS;H_123_DOCUMENTS_CLASS;H_123_DOCWINDOWS_CLASS;H_123_DRAWOBJECTS_CLASS;H_123_MAPS_CLASS;H_123_MAPBINS_CLASS;H_123_MAPTEXTENTRIES_CLASS;H_123_OLEOBJECTS_CLASS;H_123_PRINTSETTINGSCOLLECTION_CLASS;H_123_QUERYTABLES_CLASS;H_123_RANGES_CLASS;H_123_SHEETS_CLASS;H_123_STRINGS_CLASS;H_123_VERSIONS_CLASS;H_123_VERSIONGROUPS_CLASS;H_123_WINDOWS_CLASS',0)} See list of classes

{button ,AL(`H_123_NEXT_METHOD_EXSCRIPT',1)} See example

Returns the next item in a collection.

Syntax

variant **=** *object***.Next**

Parameters

None

Return values

Variant. The next item in the collection *object*. If there are no more items, 1-2-3 displays a dialog box error notification.

Usage

You can use this method in OLE automation to enumerate a collection of variant types.

By iterating no more than *object*.Count times, you can avoid the error dialog box displayed when you attempt to retrieve more items than there are in the collection.

{button ,AL(`H_123_OPEN_METHOD_MEMDEF;H_123_ITEM_METHOD_MEMDEF',0)} See related topics

1-2-3: NextSplit method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_NEXTSPLIT_METHOD_EXSCRIPT',1)} See example

Deactivates the current pane and activates the next pane, in a workbook window that is split into several panes. This method does nothing if the window is not split.

Syntax

document.**NextSplit**

Parameters

None

Return values

None

Usage

You can use this method to implement the F6 (PANE) key.

{button ,AL(`H_123_VIEWSPLITSTYLE_PROPERTY_MEMDEF;H_123_CLEARSPLITS_METHOD_MEMDEF',0)} See related topics

1-2-3: Open method

{button ,AL(`H_123_BASECOLLECTION_CLASS;H_123_CHARTS_CLASS;H_123_COLORS_CLASS;H_123_DATALINKS_CLASS;H_123_DOCUMENTS_CLASS;H_123_DOCWINDOWS_CLASS;H_123_DRAWOBJECTS_CLASS;H_123_MAPS_CLASS;H_123_MAPBINS_CLASS;H_123_MAPTEXTENTRIES_CLASS;H_123_OLEOBJECTS_CLASS;H_123_PRINTSETTINGSCOLLECTION_CLASS;H_123_QUERYTABLES_CLASS;H_123_RANGES_CLASS;H_123_SHEETS_CLASS;H_123_STRINGS_CLASS;H_123_VERSIONS_CLASS;H_123_VERSIONGROUPS_CLASS;H_123_WINDOWS_CLASS',0)} See list of classes

{button ,AL(`H_123_NEXT_METHOD_EXSCRIPT',1)} See example

Initializes a collection for item retrieval and returns the first item in the collection.

Syntax

item **=** *object***.Open**

Parameters

None

Return values

Variant. The first item in the collection *object*.

Usage

You can use this method in OLE automation to begin enumerating a collection of variant types.

{button ,AL(`H_123_NEXT_METHOD_MEMDEF;H_123_ITEM_METHOD_MEMDEF',0)} See related topics

1-2-3: OpenDocument method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENDOCUMENT_METHOD_EXSCRIPT',1)} See example

Opens an existing file as a 1-2-3 document (workbook file) in this application.

Syntax

Set *document* **=** *application*.**OpenDocument**(*filename,* [*location*], [*filetype*], [*password*], [*readonly*], [*makevisible*], [*addtorecentfiles*], [*adjacentdocument*], [*openmode*])

Parameters

filename

String. The name of the file, or the path and name of the file. If you only supply the filename here, 1-2-3 looks for the file in the working directory.

location

(Optional) Variant (String). The path where the file is located. 1-2-3 only accepts strings here.

filetype

(Optional) String. The file format to be used. If you don't supply this argument, the file extension determines the file format. The following table lists the allowed values for this parameter.

Value	Description
"All (*)"	All file types
"1-2-3 Workbook (123;WK*)"	1-2-3 97 workbook
	1-2-3 Releases 4, 5
	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
	1-2-3 for DOS Release 2
"1-2-3 SmartMaster (12M)"	1-2-3 SmartMaster template (.12m, .wt4)
"Text (TXT;PRN;CSV;DAT;OUT;ASC)"	Text file (.txt, .prn, .csv, .dat, .out, .asc)
"Excel (XLS;XLT;XLW)"	Microsoft Excel file (.xls, .xlt, .xlw)
"Quattro Pro (WQ1;WB1;WB2)"	Quattro Pro file
"dBase (DBF)"	Borland dBASE file
"Paradox (DB)"	Borland Paradox file

password

(Optional) String. A password associated with the file.

readonly

(Optional) Variant (Boolean). Specifies whether the file is to be opened in read-only mode (value True) or read-write (value False). The default is writable (False).

makevisible

(Optional) Variant (Boolean). Specifies whether the document window should be displayed (value True) or not (value False). The default is visible (True).

addtorecentfiles

(Optional) Variant (Boolean). Specifies whether to add the file to the most recent file list (value True) or not (value False). The default is not to add it (False).

adjacentdocument

(Optional) Document. Specifies a document in memory before or after which to open the file.

openmode

(Optional) Variant (OpenMode enumeration). Specifies whether to open the document before *adjacentdocument* (value $OpenModeBefore) or after *adjacentdocument* (value $OpenModeAfter). The default is $OpenModeAfter.

Return values
Document. A new instance of the Document class for this workbook file.

Usage
1-2-3 creates a new Document object for every workbook at the time the workbook is opened.

This method can generate an error for most file-related exceptions.

{button ,AL(`H_123_CLOSE_METHOD_MEMDEF:H_123_SAVEAS_METHOD_MEMDEF;H_123_NEWDOCUMENT_METHOD_MEMDEF',0)} See related topics

1-2-3: OpenDocumentFromInternet method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENDOCUMENTFROMINTERNET_METHOD_EXSCRIPT',1)} See example

Opens a document from the Internet.

Syntax

Set *document* **=** *application*.**OpenDocumentFromInternet**([*url*], [*filetype*], [*filepassword*], [*makevisible*], [*userid*], [*userpassword*], [*passiveconnection*], [*proxyserver*], [*proxyport*], [*proxytype*])

Parameters

url

(Optional) String. The Universal Resource Locator for the document (for example, "ftp://myftpserver/users/bob/test.123"). If you omit this argument, a dialog box prompts the user for it.

filetype

(Optional) String. The file format to be used. The following table lists the allowed values for this parameter.

Value	Description
"All (*)"	All file types
"1-2-3 Workbook (123;WK*)"	1-2-3 97 workbook
	1-2-3 Releases 4, 5
	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
	1-2-3 for DOS Release 2
"1-2-3 SmartMaster (12M)"	1-2-3 SmartMaster template (.12m, .wt4)
"Text (TXT;PRN;CSV;DAT;OUT;ASC)"	Text file (.txt, .prn, .csv, .dat, .out, .asc)
"Excel (XLS;XLT;XLW)"	Microsoft Excel file (.xls, .xlt, .xlw)
"Quattro Pro (WQ1;WB1;WB2)"	Quattro Pro file
"dBase (DBF)"	Borland dBASE file
"Paradox (DB)"	Borland Paradox file

The following file types are not supported (because 123 "combines" these types with the current workbook):

Value	Description
"Windows Metafile (wmf)"	Windows metafile
"Bitmap (bmp)"	Bitmap (Paint or Paintbrush)
"Ansi Metafile (cgm)"	ANSI metafile (CGM)
"1-2-3 PIC (pic)"	1-2-3 picture

filepassword

(Optional) String. A password associated with the file.

makevisible

(Optional) Variant (Boolean). Specifies whether the document window should be opened and brought to the top (value True) or not (value False). The default is not visible (False). 1-2-3 may ignore this parameter.

userid

(Optional) String. The user login name on the remote server. If you specify this argument, you must also supply the *userpassword* and *passiveconnection* arguments.

userpassword

(Optional) String. The user login password on the remote server.

passiveconnection

(Optional) Variant (Boolean). Specifies whether a passive connection to the remote server should be used (value True) or not (value False).

proxyserver

(Optional) String. The internet proxy server IP address or domain name. If you specify this argument, you must also supply the *proxyport* and *proxytype* arguments.

proxyport

(Optional) Long. The port number to use to connect to the proxy server.

proxytype

(Optional) Long. The proxy type. The following table lists the allowed values for this parameter.

Value	Description
1	World Wide Web
2	File Transfer Protocol (FTP)

Return values

Document. A new instance of the Document class for this Internet file.

{button ,AL(`H_123_SAVEASTOINTERNET_METHOD_MEMDEF;H_123_SETINTERNETOPTIONS_METHOD_MEMDEF;H_123_RETRIEVEFILEFROMINTERNET_METHOD_MEMDEF',0)} See related topics

1-2-3: OpenDocumentFromNotes method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENDOCUMENTFROMNOTES_METHOD_EXSCRIPT',1)} See example

Opens a file attached to a Notes document.

Syntax

Set *document* **=** *application*.**OpenDocumentFromNotes**([*attachedfilename*], [*universalnotesid*], [*fieldname*], [*databasefile*], [*servername*], [*filetype*], [*docpassword*], [*makevisible*])

Parameters

attachedfilename

(Optional) String. The name of the attached file in the Notes document (for example, "test.123"). If you omit this argument, a dialog box prompts the user for it. If you specify this argument, you must also specify *universalnotesid, fieldname, databasefile,* and *servername*.

universalnotesid

(Optional) String. The 32-character hexadecimal Notes document ID (the NotesDocument.UniversalID property). For example, "150DFE45F1089B790065828D852562CA".

fieldname

(Optional) String. The name of the field in which the file is attached (for example, "Body").

databasefile

(Optional) String. The Notes database location (for example, "Databases\Docs in Progress.nsf").

servername

(Optional) String. The Notes server name (for example, "Local").

filetype

(Optional) String. The file format to be used. The following table lists the allowed values for this parameter.

Value	Description
"All (*)"	All file types
"1-2-3 Workbook (123;WK*)"	1-2-3 97 workbook
	1-2-3 Releases 4, 5
	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
	1-2-3 for DOS Release 2
"1-2-3 SmartMaster (12M)"	1-2-3 SmartMaster template (.12m, .wt4)
"Text (TXT;PRN;CSV;DAT;OUT;ASC)"	Text file (.txt, .prn, .csv, .dat, .out, .asc)
"Excel (XLS;XLT;XLW)"	Microsoft Excel file (.xls, .xlt, .xlw)
"Quattro Pro (WQ1;WB1;WB2)"	Quattro Pro file
"dBase (DBF)"	Borland dBASE file
"Paradox (DB)"	Borland Paradox file

The following file types are not supported (because 123 "combines" these types with the current workbook):

Value	Description
"Windows Metafile (WMF)"	Windows metafile
"Bitmap (BMP)"	Bitmap (Paint or Paintbrush)
"Ansi Metafile (CGM)"	ANSI metafile (CGM)
"1-2-3 PIC (PIC)"	1-2-3 picture

docpassword

(Optional) String. A password associated with the file.

makevisible

(Optional) Variant (Boolean). Specifies whether the document window should be opened and brought to the top (value True) or not (value False). The default is not visible (False). 1-2-3 may ignore this parameter.

Return values

Document. A new instance of the Document class for this file attachment.

{button ,AL(`H_123_SAVEASTONOTES_METHOD_MEMDEF',0)} See related topics

1-2-3: OutlineColumnsToLevel method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_OUTLINECOLUMNSTOLEVEL_METHOD_EXSCRIPT;',1)} See example

Collapses or expands columns for a sheet that are set at or above the specified outline level. For example, to display columns whose outline level is 1 or 2, but not 3, specify OutlineColumnsToLevel to 2.

Syntax

sheet.**OutlineColumnsToLevel** *level*

Parameters

level

Long. The minimum outline level to expand to in the specified sheet.

Return values

None

Usage

You can create up to 8 levels of outlining in a sheet.

{button ,AL(`;H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_OUTLINEROWSTOLEVEL_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;',0)} See related topics

1-2-3: OutlineRowsToLevel method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_OUTLINEROWSTOLEVEL_EXSCRIPT;',1)} See example

Expands or collapses rows for a sheet that are set at or above the specified outline level. For example, to display rows whose outline level is 1 or 2, but not 3, specify OutlineRowsToLevel to 2.

Syntax

sheet.**OutlineRowsToLevel** *level*

Parameters

level

Long. The minimum outline level to expand to in the specified sheet.

Return values

None

Usage

You can create up to 8 levels of outlining in a sheet.

{button ,AL(`;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_OUTLINECOLUMNSTOLEVEL_METHOD_MEMDEF;;H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;',0)} See related topics

1-2-3: PageBack method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_PAGEBACK_METHOD_EXSCRIPT ',1)} See example

Displays a previous sheet in the current file.

Syntax

document.**PageBack** [*sheetcount*]

Parameters

sheetcount

(Optional) Long. The number of sheets to move backwards.

Return values

None

Usage

If you specify a number in *sheetcount* that is greater than the number of sheets preceding the current sheet, PageBack displays the first sheet in the file.

{button ,AL(`H_123_PAGEFORWARD_METHOD_MEMDEF;H_123_SCROLLTOACTIVECELL_METHOD_MEMDEF; H_123_TURNTO_METHOD_MEMDEF',0)} See related topics

1-2-3: PageForward method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_PAGEFORWARD_METHOD_EXSCRIPT',1)} See example

Displays the next sheet or a subsequent sheet in the current file.

Syntax

document.**PageForward** [*sheetcount*]

Parameters

sheetcount

(Optional) Long. The number of sheets to move forward.

Return values

None

Usage

If you specify a number in *sheetcount* that is greater than the number of sheets following the current sheet, PageForward displays the last sheet in the file.

{button ,AL(`H_123_PAGEBACK_METHOD_MEMDEF;H_123_SCROLLTOACTIVECELL_METHOD_MEMDEF;H_123_TURNTO_METHOD_MEMDEF',0)} See related topics

1-2-3: Paste method

{button ,AL(`;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DOCUMENT_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_COPYTOCLIPBOARD_METHOD_EXSCRIPT ',1)} See example

Copies data and related formatting from the Clipboard to the target object.

Syntax

object.**Paste**([*format*], [*link*], [*attributes*], [*icon*], [*iconfile*], [*iconindex*], [*iconlabel*])

Parameters

format

(Optional) Variant (ClipboardFormat enumeration). The format in which to paste from the Clipboard. The following table lists the allowed values for this parameter. If you omit *format*, 1-2-3 uses all appropriate formats.

Value	Description
$NativeFormat	Native format
$RichTextFormat	Rich text format
$BitmapFormat	Bitmap format
$PictureFormat	Picture format
$LotusChartFormat	Lotus Chart format
$EmbedSourceFormat	Embed source format
$EmbeddedObjectFormat	Embedded object format
$DIBFormat	Device-independent bitmap
$TextFormat	Text
$WK1Format	1-2-3 for DOS, Release 2
$WK3Format	1-2-3 for Windows Release 1; 1-2-3 for DOS Releases 3 and 4
$LinkSourceFormat	Link source format

link

(Optional) Variant (Boolean). Specify True if the data on the Clipboard is linked to an external file, False if not. The default is False.

attributes

(Optional) Variant (PasteSpecialChoice enumeration). The format in which to paste Clipboard data that was copied or cut from 1-2-3. The following table lists the allowed values for this parameter.

Value	1-2-3 pastes
$PasteData	Cell contents, but leaves the styles in the target object intact; default if you omit the parameter
$PasteStyleAndNumberFormats	All formatting done with the InfoBox
$PasteComments	Comments attached to the target object
$PasteScripts	Scripts associated with the target object
$PasteFormulas	Both cell contents and

	styles, but converts all formulas to values
$PasteBorders	Object borders
$PastesDrawObjectsWithRange	Any graphic objects associated with the 1-2-3 range on the Clipboard

icon

(Optional) Variant (Boolean). Specify True to display the Clipboard data as an icon. The default is False.

iconfile

(Optional) String. If *icon* is True, specifies the name of the file containing the icon bitmap to display.

iconindex

(Optional) Long. Specifies the index for the icon bitmap to use if the *iconfile* contains more than one icon bitmap.

iconlabel

(Optional) String. A description that is displayed below the icon bitmap.

Return value

None

{button ,AL(`;H_123_CLEAR_METHOD_MEMDEF;H_123_COPYTOCLIPBOARD_METHOD_MEMDEF;H_123_CUT_METHOD_MEMDEF;H_123_QUICKCOPY_METHOD_MEMDEF;H_123_QUICKMOVE_METHOD_MEMDEF',0) } See related topics

1-2-3: Preview method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_PREVIEW_METHOD_EXSCRIPT',1)} See example

Displays the print Preview window and InfoBox, using the current PrintSettings object for the current document. If a range is selected, just that range is previewed. Otherwise, the current sheet is previewed.

Syntax

application.**Preview**

Parameters

None

Return values

None

1-2-3: Print method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_PRINT_METHOD_EXSCRIPT;H_123_SHOWPAGEBREAKS_PROPERTY_EXSCRIPT;',1)} See example

Starts a print operation, using the current print settings object for the current document.

Syntax

application.**Print**

Parameters

None

Return values

None

Usage

Use the PrintOut method when writing scripts in applications that reserve Print as a keyword.

{button ,AL(`H_123_PRINTOUT_METHOD_MEMDEF;H_123_PRINTTOFILE_METHOD_MEMDEF',0)} See related topics

1-2-3: PrintOut method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Starts a print operation, using the current print settings object for the current document.

Syntax

application.**PrintOut**

Parameters

None

Return values

None

Usage

Use the PrintOut method when writing scripts in applications that reserve Print as a keyword.

{button ,AL(`H_123_PRINT_METHOD_MEMDEF;H_123_PRINTTOFILE_METHOD_MEMDEF',0)} See related topics

1-2-3: PrintToFile method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Displays the Print To File dialog box (as obtained from the File - Print menu).

Syntax

application.**PrintToFile**

Parameters

None

Return values

None

Usage

Use this method to allow the user to select a path and file name for exporting the workbook data as a text (.PRN) file.

{button ,AL(`H_123_PRINTOUT_METHOD_MEMDEF;H_123_PRINT_METHOD_MEMDEF',0)} See related topics

1-2-3: PromoteColumn method

{button ,AL(`H_123_Range_Class;',0)} See list of classes

{button ,AL(`H_123_PROMOTECOLUMN_METHOD_EXSCRIPT ',1)} See example

Moves the specified columns up one or more outline levels.

Syntax

range.**PromoteColumn** [*levels*]

Parameters

levels

(Optional) Long. The number of levels to promote from the current level. The default is 1 (one) level. The maximum is 8 levels.

Return values

None

Usage

The PromoteColumn method attempts to promote each column in the selection or in the specified range, by one outline level (the default) or by the optionally specified number of levels. If a column in the selection cannot be promoted because it is currently at the top level or its child would be more than one level removed from its new level, the PromoteColumn method does not promote that column.

This method does not work on 3D ranges.

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_COLLAPSEALL_METHOD_MEMDEF;H_123_EXPANDALL_METHOD_MEMDEF',0)} See related topics

1-2-3: PromoteRow method

{button ,AL(`H_123_Range_Class;',0)} See list of classes

{button ,AL(`H_123_PROMOTEROW_METHOD_EXSCRIPT ',1)} See example

Moves the specified rows up one or more outline levels.

Syntax

range.**PromoteRow** [*levels*]

Parameters

levels

(Optional) Long. The number of levels to promote from the current level. The default is 1 (one) level. The maximum is 8 levels.

Return values

None

Usage

The PromoteRow method attempts to promote each row in the selection or in the specified range, by one outline level (the default) or by the optionally specified number of levels. If a row in the selection cannot be promoted because it is currently at the top level or its child would be more than one level removed from its new level, the PromoteRow method does not promote that row.

This method does not work on 3D ranges.

{button ,AL(`H_123_DEMOTECOLUMN_METHOD_MEMDEF;H_123_PROMOTECOLUMN_METHOD_MEMDEF;H_123_COLLAPSECOLUMN_METHOD_MEMDEF;H_123_EXPANDCOLUMN_METHOD_MEMDEF;H_123_DEMOTEROW_METHOD_MEMDEF;H_123_PROMOTEROW_METHOD_MEMDEF;H_123_COLLAPSEROW_METHOD_MEMDEF;H_123_EXPANDROW_METHOD_MEMDEF;H_123_COLLAPSEALL_METHOD_MEMDEF;H_123_EXPANDALL_METHOD_MEMDEF',0)} See related topics

1-2-3: QuerySortDefineKey method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_QUERYSORTDEFINEKEY_METHOD_EXSCRIPT',1)} See example

Defines the column to sort by for a query.

Syntax

query.**QuerySortDefineKey** *fieldname*, *sortcolumn*, *sortdirection*

Parameters

fieldname

String. The name of the field in the query to sort on. For example, "ZipCodes."

sortcolumn

Long. The sort key number, from 1 - 255. This must be sequential starting at 1.

sortdirection

Variant (Direction enumeration). The direction of the sort. The following table lists the allowed values for this parameter.

Value	Description
$Ascend	Sorts from A - Z or from 0 (zero) to the highest number.
$Descend	Sorts from Z - A or from the highest number to 0 (zero).

Return values

None

{button ,AL(`;H_123_SORT_METHOD_MEMDEF;H_123_SORTRANGE_PROPERTY_MEMDEF;H_123_SORTDATA_METHOD_MEMDEF',0)} See related topics

1-2-3: QuickCopy method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

Copies data and related styles and number formats from the source range to the destination range, without using the Clipboard.

Syntax

range.**QuickCopy** *destinationrange,* [*noborders*]

Parameters

destinationrange

Variant. The name or address of the range to copy to. *Destinationrange* must be read-write.

noborders

(Optional) Variant (Boolean). Specify True to copy the content and styles from the source range, but not the borders, False to copy content, styles, and borders. The default is False.

Usage

If you omit *range*, 1-2-3 uses the current selection as the source range.

Return values

None

{button ,AL(`H_123_QUICKMOVE_METHOD_MEMDEF',0)} See related topics

1-2-3: QuickMove method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

Moves data and related styles and number formats from the source range to the destination range, without using the Clipboard.

Syntax

range.**QuickMove** *destinationrange,* [*noborders*]

Parameters

destinationrange

Variant. The name or address of the range to copy to. *Destinationrange* must be read-write.

noborders

(Optional) Variant (Boolean). Specify True to copy the content and styles from the source range, but not the borders. Specify False to copy content, styles, and borders. The default is False.

Usage

If you omit *range*, 1-2-3 uses the current selection as the source range.

Return values

None

{button ,AL(`H_123_QUICKCOPY_METHOD_MEMDEF;',0)} See related topics

1-2-3: Quit method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_QUIT_METHOD_EXSCRIPT',1)} See example

Exits the application. Saves open documents if so specified by the *savechanges* argument.

Syntax

application.**Quit** *savechanges*

Parameters

savechanges

Variant (Boolean). Specifies whether to save changes to open files (value True) or not (value False). If you specify False for *savechanges*, 1-2-3 closes changed files without saving, and without asking the user whether to save them. If you omit the *savechanges* argument, a dialog box appears when there are changed files to be closed, asking the user whether to save them.

Return values

None

{button ,AL(`H_123_CLOSE_METHOD_MEMDEF;H_123_NEWDOCUMENT_METHOD_MEMDEF;H_123_OPEN_METHOD_MEMDEF;H_123_OPENDOCUMENT_METHOD_MEMDEF;H_123_CLOSEALL_METHOD_MEMDEF',0)} See related topics

1-2-3: RangeCombine method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_RANGECOMBINE_METHOD_EXSCRIPT',1)} See example

Combines data and number formats from a range in a file on disk into the current workbook.

Syntax

range.**RangeCombine** *filename,* [*location*], [*filetype*], [*password*], [*combineoption*], [*sourcerange*]

Parameters

filename

String. Specifies the name of the file on disk containing data which you want to combine with data in the current workbook.

location

(Optional) String. Specifies the location of the file containing the data you want to combine. If you omit this parameter, 1-2-3 uses the working directory as the location for the input file.

filetype

(Optional) String. Specifies the file format. The following table lists the allowed values for this parameter. If you omit this parameter, 1-2-3 uses the file extension to determine the file format.

Value	Description
"All (*)"	All file types
"1-2-3 Workbook (123;WK*)"	1-2-3 workbook
"1-2-3 SmartMaster (12M)"	1-2-3 SmartMaster template
"Text (TXT;PRN;CSV;DAT;OUT;ASC)"	Text
"DBase (DBF)"	Borland dBASE
"Paradox (DB)"	Borland Paradox

password

(Optional) String. A password associated with the file.

combineoption

(Optional) Variant (WALCombineOption enumeration). Specifies how you want 1-2-3 to combine data. You can only specify this argument if you are combining data from 1-2-3 files. The following table lists the allowed values for this parameter.

Value	Description
$CombineAdd	Adds numbers and the results of numeric formulas in a file on disk to numbers or blank cells in the current file.
$CombineReplace	Replaces data in the current file with data copied from the file on disk; default if you omit the parameter.
$CombineSub	Subtracts numbers and the results of numeric formulas in a file on disk from numbers or blank cells in the current file.

sourcerange

(Optional) String. The name or address of the range in *filename* that contains data that you want to combine with data in the current file. If you omit *sourcerange*, 1-2-3 combines all the data in *filename* with data in the current file.

Return values

None

1-2-3: RangeCombineText method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_RANGECOMBINETEXT_METHOD_EXSCRIPT',1)} See example

Combines data from a text file into the specified file in the current workbook.

Syntax

range.**RangeCombineText**(*inputfilename,* [*path*], [*readtextas*], [*delimiter*], [*characterset*])

Parameters

inputfilename

String. Specifies the name of the file on disk containing data which you want to combine with data in the current workbook.

path

(Optional) String. Specifies the location of the file containing the data you want to combine.

readtextas

(Optional) Variant (enumeration). Specifies how 1-2-3 should combine data from a text file. The following table lists the allowed values for this parameter. The values are from the ReadTestAs enumeration.

Value	Description
$AutoParse	1-2-3 automatically parses the text file by determining where the breaks are, then breaks the data into separate columns in the sheet.
$Tab	Use tabs as delimiters.
$Comma	Use commas as delimiters.
$Semicolon	Use semicolons as delimiters.
$Space	Use spaces as delimiters.
$Other	Use the character specified by *delimiter* to parse the text file.
$None	1-2-3 enters each line of data as a label in a separate cell using successive cells in the same column.
$Text	1-2-3 enters each line of data as a label in a separate cell using successive cells in the same column.
$Numbers	1-2-3 enters labels and numbers from a delimited text file and enters them in a separate cell using successive cells in the same column. When a text file is not delimited, 1-2-3 enters only numbers.

delimiter

String. The delimiter character to use when $Other is specified as the *readtextas* option.

characterset

Variant (enumeration). Specifies the code page you want 1-2-3 to use for interpreting the data in the text file. The following table lists the allowed values for this parameter. The values are from the ReadCharSet enumeration.

Value	Description

$Windows	Windows ANSI; default if you omit this parameter
$DOS	DOS or OS/2
$CP1252	Multilingual Windows
$CP850	Multilingual DOS
$CP932	Japanese
$BIG5	Taiwanese
$KS	Korean
$GB	Chinese
$CP1252	US Windows
$CP437	US DOS
$CP860	Portuguese DOS
$CP863	French Canadian DOS
$CP865	Nordic DOS
$CP1250	Eastern European Windows
$CP852	Eastern European DOS
$CP1251	Cyrillic Windows
$CP866	Cyrillic DOS
$CP1253	Greek Windows
$CP851	Greek DOS

Return values
None

Usage
Make sure that numbers in a text file do not contain commas, since these are delimiters. For example, 1-2-3 interprets 12,345 as two values, 12 and 345.

1-2-3: RangeExtract method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_RANGEEXTRACT_METHOD_EXSCRIPT',1)} See example

Saves a range to another file.

Syntax

range.**RangeExtract** *filename,* [*location*], [*filetype*], [*backup*], [*password*], [*extractoption*]

Parameters

filename

String. Specifies the name of the file to which you want to save the range.

location

(Optional) String. Specifies the location of the file to which you want to save the range. The default is the working directory.

filetype

(Optional) String. Specifies the format of the saved file. The following table lists the allowed values for this parameter. If you omit this parameter, 1-2-3 uses the file extension to determine the file format.

Value	Description
"1-2-3 (123)"	1-2-3 97 workbook
"1-2-3 (WK4)"	1-2-3 Release 4
"1-2-3 (WK3)"	1-2-3 Release 3
"1-2-3 (WK1)"	1-2-3 for DOS Release 2
"SmartMaster (12M)"	1-2-3 SmartMaster template
"Text (TXT)"	Text
"DBase (DBF)"	Borland dBASE
"Paradox (DB)"	Borland Paradox

backup

(Optional) Variant (boolean). Specifies whether or not to create a backup file if *filename* specifies an existing file. The default is False.

password

(Optional) String. A password associated with the file.

If you omit *password*, and the file specified by *filename* has a password, 1-2-3 saves the file without a password.

extractoption

(Optional) Variant (enumeration). Specifies how to save formulas in the range to the file. The following table lists the allowed values for this parameter. The values are from the WALExtractOption enumeration.

Value	Description
$ExtractValuesAndFormulas	Saves formulas without converting them to values; default if you omit this parameter.
$ExtractValuesOnly	Converts formulas to values.

Return values

None

1-2-3: RangeFill method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RANGEFILL_METHOD_EXSCRIPT',1)} See example

Enters a sequence of values in a range.

Syntax

range.**RangeFill**([*startvalue*], [*increment*], [*stopvalue*], [*filltype*], [*datetimeunits*])

Parameters

startvalue

(Optional) Variant. Specifies the first value 1-2-3 enters in the range.

increment

(Optional) Variant. Specifies the increment between each of the values in the range.

stopvalue

(Optional) Variant. Specifies the limit of the sequence. If you specify a negative *increment*, you must specify a *stopvalue* that is less than the *startvalue*.

filltype

(Optional) Variant (enumeration). Specifies the type of values to fill. The following table lists the allowed values for this parameter. The values are from the WorksheetFillType enumeration.

Value	Description
$Number	Fills the range with a sequence of numbers; default if you omit this parameter.
$DateValue	Fills the range with a sequence of dates.
$TimeValue	Fills the range with a sequence of times.
$FillByExample	Fills the range with a sequence, based on data you include in the range.

datetimeunits

(Optional) Variant (enumeration). When *filltype* is $DateValue or $TimeValue, specifies the step increment as a unit of time. The following table lists the allowed values for this parameter. The values are from the FillInterval enumeration.

Value	Description
$Day	Day
$Week	Week (7 days)
$Weekday	Work week (5 days)
$Month	Month (30 or 31 days)
$Quarter	Quarter (90 days)
$Year	Year (365 or 366 days)
$Hour	Hour
$Minute	Minute
$Second	Second

Return values

None

Usage

If you omit *startvalue, increment,* or *stopvalue*, 1-2-3 uses the same values as the last time you used RangeFill during the current 1-2-3 session.

1-2-3: RangeSortDefineKey method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_RANGESORTDEFINEKEY_METHOD_EXSCRIPT',1)} See example

Defines the column by which to sort the range of data.

Syntax

document.**RangeSortDefineKey** *keynumber, keyrange, sortdirection*

Parameters

keynumber

Long. Any integer from 0 through 255 that specifies a sort key.

keyrange

Variant. Specifies the name or address of the range to be sorted for this key. An empty string, "", is a legitimate input value.

sortdirection

Variant (enumeration). Specifies the sort order for *keyrange.* The following table lists the allowed values for this parameter. The values are from the SortDir enumeration.

Value	Description
$Ascend	Sorts from A - Z, and smallest to largest values
$Descend	Z - A, and largest to smallest values

Return values

None

1-2-3: RangeValue method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RANGEVALUE_METHOD_EXSCRIPT',1)} See example

Copies the contents and styles from a range, and replaces all copied formulas with their current values.

Syntax

range.**RangeValue**(*destinationrange*)

Parameters

destinationrange

Variant. The name or address of the range to which you are copying. Specify either the entire range or only the first cell.

Return values

None

1-2-3: RecalcRange method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_RECALCRANGE_METHOD_EXSCRIPT',1)} See example

Specifies the recalculation order and then recalculates a range.

Syntax

range.**RecalcRange** (*recalcorder*)

Parameters

recalcorder

Variant (RecalcOrder enumeration). Specifies how 1-2-3 recalculates the range. The following table lists the allowed values for this parameter.

Value	Description
$Columns	Recalculates all formulas starting in the first cell of the range and moves column by column through each sheet in the range.
$Rows	Recalculates all formulas starting in the first cell of the range and moves row by row through each sheet in the range.

Return values

None

1-2-3: RecenterMap method

{button ,AL(`H_123_MAP_CLASS;',0)} See list of classes

Specifies the latitude and longitude for the center of the plot area of a map.

Syntax

map.**RecenterMap**(*latitude, longitude*)

Parameters

latitude

Double. The latitude, in degrees, you want to display in the center of the plot area.

longitude

Double. The longitude, in degrees, you want to display in the center of the plot area.

Return values

None

{button ,AL(`;H_123_REDRAWMAP_METHOD_MEMDEF;H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_ZOOMMAPIN_METHOD_MEMDEF;H_ZOOMMAPOUT_METHOD_MEMDEF;H_ZOOMMAPRESET_METHOD_MEMDEF;H_ZOOMMAPTO_METHOD_MEMDEF;H_ZOOMMAPTORECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: RedefineNamedPrintSettings method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Redefines the specified named print settings (print style) by applying the settings in the document's CurrentPrintSettings property to the named print settings.

Syntax

document.**RedefineNamedPrintSettings** *settingsname, printrange*

Parameters

settingsname

String. The name of the PrintSettings object to be redefined.

printrange

Variant (Boolean). If True, the currently selected range is used as the print range in the new print settings. If False, the print range in the CurrentPrintSettings property is used.

Return values

None

{button ,AL(`H_123_DELETENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NEWNAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NAMEDPRINTSETTINGS_PROPERTY_MEMDEF',0)} See related topics

1-2-3: RedrawMap method

{button ,AL(`H_123_MAP_CLASS;',0)} See list of classes

Redraws a map.

Syntax

*map.***RedrawMap**

Parameters

None

Return values

None

{button ,AL(`H_123_RECENTERMAP_METHOD_MEMDEF;H_123_REDRAWMAP_METHOD_MEMDEF;H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_ZOOMMAPIN_METHOD_MEMDEF;H_ZOOMMAPOUT_METHOD_MEMDEF;H_ZOOMMAPRESET_METHOD_MEMDEF;H_ZOOMMAPTO_METHOD_MEMDEF;H_ZOOMMAPTORECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: Refresh method

{button ,AL(`H_123_REFRESH_METHOD_MEMDEF_RT;H_123_QUERY_CLASS;',0)} See list of classes

{button ,AL(`H_123_REFRESH_METHOD_EXSCRIPT',1)} See example

Runs a query and refreshes the output range with the new result records.

Syntax

query.**Refresh**

Parameters

None

Return values

None

{button ,AL(`;H_123_SELECT_METHOD_MEMDEF;H_123_UPDATE_METHOD_MEMDEF',0)} See related topics

1-2-3: RefreshOutput method

{button ,AL(`H_REFRESHOUTPUT_METHOD_MEMDEF_RT;H_123_QUERYTABLE_CLASS;',0)} See list of classes

{button ,AL(`H_123_REFRESHOUTPUT_METHOD_EXSCRIPT',1)} See example

Refreshes the output range based upon the current query table. This makes the contents of the output range identical to those in the query table. The query table is not updated.

Syntax

*querytable.***RefreshOutput**

Parameters

None

Return values

None

Usage

The RefreshOutput method is useful once you are satisfied with the contents of your query table and want to copy output from the query table to the sheet.

{button ,AL(`;H_123_REFRESHQUERY_METHOD_MEMDEF;H_123_UPDATE_METHOD_MEMDEF',0)} See related topics

1-2-3: RefreshQuery method

{button ,AL(`H_REFRESHQUERY_METHOD_MEMDEF_RT;H_123_QUERYTABLE_CLASS;',0)} See list of classes

{button ,AL(`H_123_REFRESHOUTPUT_METHOD_EXSCRIPT',1)} See example

Refreshes the contents of the query table. This method automatically performs a RefreshOutput, and the output range is refreshed to match the contents of the query table.

Syntax

querytable.**RefreshQuery**

Parameters

None

Return values

None

Usage

The RefreshQuery method is useful after you have modified your query and want to refresh data in both the query table and its output range.

If the query table is based on data that is external to 1-2-3, you can use the RefreshQuery method to update your query table and output with the latest data in the source database.

{button ,AL(`;H_123_REFRESHOUTPUT_METHOD_MEMDEF;H_123_UPDATE_METHOD_MEMDEF',0)} See related topics

1-2-3: Regression method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Performs multiple linear regression analysis and also calculates the slope of the line that best illustrates the data.

Syntax

document.**Regression** *xrange, yrange, outputrange, intercept*

Parameters

xrange

Variant. Contains the independent variables. *Xrange* is the name or address of a range that can contain up to 75 columns and 8,192 rows.

yrange

Variant. Contains the set of values for the dependent variable. *Yrange* is the name or address of a single-column range with the same number of rows as *xrange*.

outputrange

Variant. Specifies the name or address of a range for the results of the regression analysis. Specify either the entire range or only the first cell.

Caution 1-2-3 writes over any existing data in *outputrange*.

intercept

Variant (enumeration). Specifies whether 1-2-3 calculates the y-axis intercept or uses 0 as the y-axis intercept. The following table lists the allowed values for this parameter. The values are from the RegressionEnum enumeration.

Value	Description
$Compute	Calculates the y-axis intercept.
$Zero	Uses 0 (zero) as the y-axis intercept.

Return values

None

1-2-3: RegressionReset method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Resets the x-range, y-range, output range, and y-axis intercept settings for a regression analysis.

Syntax

document.**RegressionReset**

Parameters

None

Return values

None

1-2-3: Remove method

{button ,AL(`H_123_DRAWCOLLECTION_CLASS',0)} See list of classes

Removes the specified object from the set of currently selected drawn objects.

Syntax

.Remove *drawobject*

Parameters

drawobject

Variant. The drawn object to be deselected. This must be a selected object derived from DrawObject.

Return values

None

{button ,AL(`H_123_ADDTOSELECTION_METHOD_MEMDEF;H_123_SELECTION_PROPERTY_MEMDEF;H_123_DRAWCOLLECTION_CLASS;H_123_REMOVEFROMSELECTION_METHOD_MEMDEF;H_123_SELECT_METHOD_MEMDEF;H_123_GROUP_METHOD_MEMDEF;H_123_UNGROUP_METHOD_MEMDEF',0)} See related topics

1-2-3: RemoveFromSelection method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_QUERY_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_REMOVEFROMSELECTION_METHOD_EXSCRIPT',1)} See example

Removes the object from the selection, while leaving other objects in the selection unchanged.

Syntax

object.**RemoveFromSelection**

Parameters

None

Return values

None

1-2-3: RemoveItem method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

Removes an item, including a submenu item, from a menu or menu bar.

Syntax

object.**RemoveItem** *position*

Parameters

position

Long. The position in the menu of the item to be removed.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position.

Return values

None

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_DISABLEITEM_METHOD_MEMDEF;H_123_REPLACE ITEM_METHOD_MEMDEF',0)} See related topics

1-2-3: RemoveOverlay method

{button ,AL(`H_123_MAP_CLASS;',0)} See list of classes

Removes an overlay that was added to a map.

Syntax

map.**RemoveOverlay**(*overlayfilename*)

Parameters

overlayfilename

String. The name of the overlay file to remove.

Return values

None

{button ,AL(`H_123_RECENTERMAP_METHOD_MEMDEF;H_123_REDRAWMAP_METHOD_MEMDEF;H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_ZOOMMAPIN_METHOD_MEMDEF;H_ZOOMMAPOUT_METHOD_MEMDEF;H_ZOOMMAPRESET_METHOD_MEMDEF;H_ZOOMMAPTO_METHOD_MEMDEF;H_ZOOMMAPTORECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: RemoveSelectField method

{button ,AL(`H_123_REMOVESELECTFIELD_METHOD_MEMDEF_RT;H_123_QUERY_CLASS;',0)} See list of classes

{button ,AL(`H_123_REMOVESELECTFIELD_METHOD_EXSCRIPT ',1)} See example

Removes the specified field from the query.

Syntax

query.**RemoveSelectField** *fieldname*

Parameters

fieldname

String. The name of the field to remove from the query, For example, "ZipCodes."

Return values

None

{button ,AL(`H_123_ADDSELECTFIELD_METHOD_MEMDEF;',0)} See related topics

1-2-3: RemoveVersion method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_REMOVEVERSION_METHOD_EXSCRIPT ',1)} See example

Removes a version from the version group.

Syntax

versiongroup.**RemoveVersion** (*rangename*)

Parameters

rangename

String. The name of the range that contains the version that you want to remove from the version group.

Return values

None

{button ,AL(`H_123_ADDVERSION_METHOD_MEMDEF;H_123_DELETEVERSION_METHOD_MEMDEF',0)} See related topics

1-2-3: RenameNamedPrintSettings method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Renames the specified named print settings (print style).

Syntax

document.**RenameNamedPrintSettings** *oldsettingsname, newsettingsname*

Parameters

oldsettingsname

String. The existing name of the PrintSettings object to be renamed.

newsettingsname

String. The new name for the PrintSettings object.

Return values

None

{button ,AL(`H_123_DELETENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NEWNAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_REDEFINENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_RETRIEVENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NAMEDPRINTSETTINGS_PROPERTY_MEMDEF;H_123_CURRENTPRINTSETTINGS_PROPERTY_MEMDEF',0)} See related topics

1-2-3: RenameNamedStyle method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RENAMENAMEDSTYLE_METHOD_EXSCRIPT ',1)} See example

Renames a named style.

Syntax

RenameNamedStyle *"oldstylename", "newstylename"*

Parameters

oldstylename

String. The previous name of the named style.

newstylename

String. The proposed name for the named style.

Return value

None

{button ,AL(`;H_123_MODIFYNAMEDSTYLE_METHOD_MEMDEF;H_123_RENAMENAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTONAMEDSTYLE_METHOD_MEMDEF;H_123_STYLEFONTRESET_METHOD_MEMDEF;H_123_STYLENAME_PROPERTY_MEMDEF;H_123_STYLESOURCE_PROPERTY_MEMDEF',0)} See related topics

1-2-3: Replace method

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_REPLACE_METHOD_EXSCRIPT',1)} See example

Finds the first occurence of specified characters in labels, numbers, and formulas, and replaces them. You can find the nth occurrence by using the optional *occurrence* argument.

Syntax

boolean = *document.object*.**Replace** [(*occurrence*)]

Parameters

occurrence

(Optional) Integer. Specifies which occurrence of the target string to find. The default is 1.

Return value

Variant. A Boolean value indicating whether an occurrence of the target string was found.

Usage

This method searches for the string stored in the SearchString property, uses the replacement text stored in the ReplaceString property, and uses the search matching option properties also stored in the Application object.

{button ,AL(`;H_123_REPLACEALL_METHOD_MEMDEF',0)} See related topics

1-2-3: ReplaceAll method

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_REPLACE_METHOD_EXSCRIPT',1)} See example

Finds all occurrences of specified characters in labels, numbers, and formulas, and replaces them.

Syntax

boolean **=** *document.object*.**ReplaceAll**

Parameters

None

Return value

Variant. A Boolean value indicating whether an occurrence of the target string was found.

Usage

This method searches for the string stored in the SearchString property, uses the replacement text stored in the ReplaceString property, and uses the search matching option properties also stored in the Application object.

{button ,AL(`H_123_REPLACE_METHOD_MEMDEF',0)} See related topics

1-2-3: ReplaceItem method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

Replaces a non-submenu item in a menu or menu bar and binds a specified script to it. The item script is executed when the user chooses the item.

Syntax

object.**ReplaceItem** *position, menutext, menuprompt, scriptdocument, scriptname*

Parameters

position

Long. The position in the menu of the item to be replaced.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position.

menutext

String. Text for the new item to display in the menu. An & (ampersand) before any letter specifies the shortcut key for the item.

menuprompt

String. The long prompt for the item.

scriptdocument

Document. The file containing the global script to be called by the item. This must be either a reference to a Document object in the form [*docname*.123] or the predefined global product variable ThisDocument. For example, [mysheet.123].

scriptname

String. The name of the global script or sub to call when the user chooses the menu item. This script can employ the full set of 1-2-3 methods and properties, as well as LotusScript language elements.

Return values

None

Usage

To replace a submenu in a menu or menu bar, use the ReplaceMenu method.

If the item is a setting, you can indicate the status of the setting by calling the CheckItem or UncheckItem method in the script function for the item.

{button ,AL(`H_123_ADDITEM_METHOD_EXSCRIPT;H_123_CHECKITEM_METHOD_MEMDEF;H_123_DISABLEITEM_METHOD_MEMDEF;H_123_ENABLEITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_UNCHECKITEM_METHOD_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF;',0)} See related topics

1-2-3: ReplaceMenu method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

Replaces a submenu in a menu, or a menu in a menu bar.

Syntax

object.**ReplaceMenu** *position, newmenu*

Parameters

position

Long. The menu position of the menu to be replaced.

Value	Description
Positive integer	The menu's position in the menu, counting forward from the beginning. The value 1 means the first position. If *position* exceeds the position of the last existing submenu or item, the new menu is placed at the end.
Negative integer	The menu's position in the menu, counting backward from the end. The value –1 means the last position.

newmenu

Menu. The new submenu to add. You can add a custom menu you created with the NewMenu method, or a system or product menu you got with the GetMenu method.

Return values

None

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_RESETMENUBAR_METHOD_MEMDEF;H_123_GETMENU_METHOD_MEMDEF;H_123_NEWMENU_METHOD_MEMDEF;H_123_NEWMENUBAR_METHOD_MEMDEF;H_123_CURRENTMENUBAR_PROPERTY_MEMDEF',0)} See related topics

1-2-3: ReportVersion method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_REPORTVERSION_METHOD_EXSCRIPT ',1)} See example

Creates a document containing a report on versions for the specified range.

Syntax

range.**ReportVersion** *versionnames*, [*formularange, includedata, includeaudit, orientation*]

Parameters

*versionname*s

String. A list of the names of versions for the specified range to be included in the report. The list is enclosed in" " (double quotes); items are separated by ; (semicolons).

Note Version names are case-sensitive.

formularange

(Optional) Variant. The address or range name containing formulas that depend on values in one or more of the specified versions. For example, if you have a formula in a cell named NETGAIN that references cells in one or more of the specified versions of a range, specify NETGAIN for *formularange*. The resulting report lists NETGAIN as dependent formula. By default, no formulas are included in the report.

includedata

(Optional) Variant (Boolean). Specifies whether the report contains data from each of the specified versions. By default, version data is included in the report.

includeaudit

(Optional) Variant (Boolean). Specifies whether audit information for each of the specified versions is included in the report. Audit information includes the name of the user who created the version, the name of the user who last modified the version, the date and time that the version was created, and the date and time that the version was modified.

orientation

(Optional) Variant (ColumnOrRow enumeration). Specifies how the report is arranged. You can use columns or rows as its orientation. The following table lists the allowed values for this parameter.

Value	Description
$Column	(Default) Format the report using columns as the orientation for information.
$Row	Format the report using rows as the orientation for information.

Return value

None

Usage

Version reports are useful in evaluating the history, contents, and effects on formulas of each version for a specified range.

{button ,AL(`;H_123_ADDVERSION_METHOD_MEMDEF;H_123_DELETEVERSION_METHOD_MEMDEF;H_123_VERSIONS_METHOD_MEMDEF',0)} See related topics

1-2-3: ReservationGet method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_RESERVATIONGET_METHOD_EXSCRIPT',1)} See example

Attempts to get the reservation for the file.

Syntax

document.**ReservationGet**

Parameters

None

Return values

None

Usage

This method displays an error dialog box if you already have the reservation for the file.

{button ,AL(`H_123_RESERVATIONRELEASED_METHOD_MEMDEF;H_123_READONLY_PROPERTY_MEMDEF', 0)} See related topics

1-2-3: ReservationReleased method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_RESERVATIONGET_METHOD_EXSCRIPT',1)} See example

Releases the file reservation if you have it.

Syntax

document.**ReservationReleased**

Parameters

None

Return values

None

Usage

This method displays an error dialog box if you don't have the file reservation.

{button ,AL(`H_123_RESERVED_METHOD_MEMDEF',0)} See related topics

1-2-3: ResetColumnWidth method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RESETCOLUMNWIDTH_METHOD_EXSCRIPT ',1)} See example

Resets the column width for the specified range to the default column width for the sheet containing the range.

Syntax

range.**ResetColumnWidth**

Parameters

None

Return values

None

Usage

The ResetColumnWidth method uses the current value of the sheet DefaultColumnWidth property to reset column widths for a range.

{button ,AL(`;H_123_DEFAULTCOLUMNWIDTH_PROPERTY_MEMDEF;H_123_RESETROWHEIGHT_METHOD_MEMDEF',0)} See related topics

1-2-3: ResetFieldAggregates method

{button ,AL(`H_123_QUERY_CLASS;',0)} See list of classes

{button ,AL(`H_123_RESETFIELDAGGREGATES_METHOD_EXSCRIPT ',1)} See example

Resets all aggregates set on all fields in a query. If you have added one or more aggregate or summary fields in your query and have defined formulas for those aggregates, you can use this method to remove the formulas for all aggregate fields.

Syntax

query.**ResetFieldAggregates**

Parameters

None

Return values

None

{button ,AL(`H_123_FIELDAGGREGATETYPE_METHOD_MEMDEF',0)} See related topics

1-2-3: ResetMenuBar method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_RESETMENUBAR_METHOD_EXSCRIPT',1)} See example

Resets the menu bar to the default menus. Removes custom menus.

Syntax

application.**ResetMenuBar**

Parameters

None

Return values

None

Usage

If you don't remove your custom menus when the file containing their menu item handlers closes, the menus will remain up, but the items won't function.

{button ,AL(`H_123_ADDMENU_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_CURRENTMENUBAR_PROPERTY_MEMDEF;H_123_PRECLOSE_EVENT_MEMDEF',0)} See related topics

1-2-3: ResetRowHeight method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RESETROWHEIGHT_METHOD_EXSCRIPT;H_123_FITTALLEST_METHOD_EXSCRIPT',1)} See example

Resets the row height for the specified range to the default row height for the sheet containing the range.

Syntax

range.**ResetRowHeight**

Parameters

None

Return values

None

Usage

The ResetRowHeight method uses the current value of the sheet DefaultRowHeight property to reset row heights for a range.

{button ,AL(`H_123_RESETCOLUMNWIDTH_METHOD_MEMDEF;H_123_DEFAULTROWHEIGHT_PROPERTY_MEMDEF',0)} See related topics

1-2-3: Reshape method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RESHAPE_METHOD_EXSCRIPT',1)} See example

Changes the coordinates of the selected range.

Syntax

range.**Reshape**(*newrange*)

Parameters

newrange

Variant. The range coordinates to use to replace the selected range.

Return values

None

1-2-3: Resize method

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_WINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_PLOT_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_QUERYTABLE_CLASS;H_123_DRAWCOLLECTION_CLASS;',0)} See list of classes

{button ,AL(`H_123_RESIZE_METHOD_EXSCRIPT',1)} See example

For objects based on the DrawObject class or window objects, sets the width and height of the object by a specified number of units.

Syntax

object.**Resize** *width*, *height*

Paramters

width

Long. The width in twips (1/1440 of an inch).

height

Long. The height in twips (1/1440 of an inch).

Return values

None

{button ,AL(`H_123_BOUNDS_METHOD_MEMDEF;H_123_MAXIMIZE_METHOD_MEMDEF;H_123_MINIMIZE_METHOD_MEMDEF;H_123_MOVE_METHOD_MEMDEF;H_123_RESTORE_METHOD_MEMDEF',0)} See related topics

1-2-3: Restore method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_WINDOW_CLASS;H_123_DOCWINDOW_CLASS;',0) } See list of classes

Restores the specified window to its original size.

Syntax

object.**Restore**

Parameters

None

Return values

None

{button ,AL(`H_123_MAXIMIZE_METHOD_MEMDEF;H_123_MINIMIZE_METHOD_MEMDEF;H_123_MOVE_METHOD_MEMDEF;H_123_RESIZE_METHOD_MEMDEF',0)} See related topics

1-2-3: RetrieveFileFromInternet method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_RETRIEVEFILEFROMINTERNET_METHOD_EXSCRIPT',1)} See example

Retrieves a document from the Internet and stores it locally as a file in the user's TEMP directory. The file is given a unique name by the operating system.

Syntax

filename = *application*.**RetrieveFileFromInternet**(*url,* [*userid*]*,* [*userpassword*]*,* [*passiveconnection*]*,* [*proxyserver*]*,* [*proxyport*]*,* [*proxytype*])

Parameters

url

String. The Universal Resource Locator for the document (for example,

"ftp://myftpserver/users/bob/test.123").

userid

(Optional) String. The user login name on the remote server. If you specify this argument, you must also supply the *userpassword* and *passiveconnection* arguments.

userpassword

(Optional) String. The user login password on the remote server.

passiveconnection

(Optional) Variant (Boolean). Specifies whether a passive connection to the remote server should be used (value True) or not (value False).

proxyserver

(Optional) String. The Internet proxy server IP address or domain name. If you specify this argument, you must also supply the *proxyport* and *proxytype* arguments.

proxyport

(Optional) Long. The port number to use to connect to the proxy server.

proxytype

(Optional) Long. The proxy type. The following table lists the allowed values for this parameter.

Value	Description
1	World Wide Web
2	File Transfer Protocol (FTP)

Return values

String. The local name of the retrieved file on the user's computer. This name is assigned by the operating system.

{button ,AL(`H_123_OPENDOCUMENTFROMINTERNET_METHOD_MEMDEF;H_123_SAVEASTOINTERNET_METHOD_MEMDEF',0)} See related topics

1-2-3: RetrievePrintSettings method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_RETRIEVEPRINTSETTINGS_METHOD_EXSCRIPT',1)} See example

Retrieves print settings from a named page settings (.AL3) file that was created in an earlier release of 1-2-3 (Release 3, 4, or 5) and applies them to the document's CurrentPrintSettings property.

Syntax

document.**RetrievePrintSettings** *settingsfile*

Parameters

settingsfile

String. The name of the 1-2-3 page settings (.AL3) file containing the print settings you want to retrieve.

Return values

None

{button ,AL(`H_123_REDEFINENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NEWNAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NAMEDPRINTSETTINGS_PROPERTY_MEMDEF',0)} See related topics

1-2-3: RevertToNamedStyle method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_REVERTTONAMEDSTYLE_METHOD_EXSCRIPT ',1)} See example

Resets style properties for the range back to those of its named style.

Syntax

range.**RevertToNamedStyle**

Parameters

None

Return value

None

Usage

The RevertToNamedStyle method is useful when you first apply a named style to a range and then further modify style properties for that range. Its style properties, therefore, are a combination of properties defined in the named style and properties that you applied locally. Use the RevertToNamedStyle method to restore all style attributes for the range to those defined in the named style.

{button ,AL(`;H_123_MODIFYNAMEDSTYLE_METHOD_MEMDEF;H_123_RENAMENAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTOSTYLE_METHOD_MEMDEF',0)} See related topics

1-2-3: RevertToStyle method

{button ,AL(`H_123_FONT_CLASS;H_123_BACKGROUND_CLASS;',0)} See list of classes

{button ,AL(`H_123_REVERTTOSTYLE_METHOD_EXSCRIPT;',1)} See example

Resets the background and font styles for a range to the style defaults.

Syntax

object.**RevertToStyle**

Parameters

None

Return value

None

{button ,AL(`;H_123_MODIFYNAMEDSTYLE_METHOD_MEMDEF;H_123_RENAMENAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTONAMEDSTYLE_METHOD_MEMDEF',0)} See related topics

1-2-3: SameColor method

{button ,AL(`H_123_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_123_SAMECOLOR_METHOD_EXSCRIPT',1)} See example

Tests if the RGB color value of a specified object matches another RGB color value.

Syntax

colorissame = *color*.**SameColor**(*othercolor*)

Parameters

othercolor

Color. A color value.

Return value

colorissame

Boolean. Indicates whether the colors are the same. The SameColor method returns True if the RGB color of the specified object and *othercolor* are the same and returns False if they are different.

Usage

SameColor is useful if you need to compare the colors of two objects in different Lotus products.

{button ,AL(`H_123_GETRGB_METHOD_MEMDEF',0)} See related topics

1-2-3: Save method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SAVE_METHOD_EXSCRIPT',1)} See example

Saves the document to its current path. If no path and file name have been specified for the document (that is, it is a new document), 1-2-3 prompts the user for the path and file name.

Syntax

document.**Save**

Parameters

None

Return values

None

{button ,AL(`H_123_SAVEAS_METHOD_MEMDEF;H_123_SAVECOPYAS_METHOD_MEMDEF;H_123_CLOSE_METHOD_MEMDEF;H_123_CLOSEALL_METHOD_MEMDEF;H_123_OPENDOCUMENT_METHOD_MEMDEF',0)} See related topics

1-2-3: SaveAs method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENDOCUMENT_METHOD_EXSCRIPT',1)} See example

Saves the document, using the specified file name and path. If you do not specify a file name, 1-2-3 prompts the user for it.

Syntax

document.**SaveAs** *name,* [*location*], [*filetype*], [*backup*], [*password*], [*addtorecentfiles*]

Parameters

name

String. The name of the file, or the name and full path of the file to be saved.

location

(Optional) String. The path where the file is to be saved.

filetype

(Optional) String. The file format to be used to save the file. The default is "1-2-3 (123)" format. The following table lists the allowed values for this parameter.

Value	Description
"1-2-3 (123)"	1-2-3 97 workbook
"1-2-3 (WK4)"	1-2-3 for Windows Release 4, 5
"1-2-3 (WK3)"	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
"1-2-3 (WK1)"	1-2-3 for DOS Release 2
"SmartMaster (12M)"	1-2-3 97 SmartMaster template
"Text (TXT)"	Text
"Excel Worksheet (XLS)"	Microsoft Excel worksheet
"Excel Workbook (XLW)"	Microsoft Excel workbook

backup

(Optional) Variant (Boolean). Specifies whether to back up the file (value True) or not (value False). If you specify False for this argument, 1-2-3 replaces the file previously saved with the one you are saving.

password

(Optional) String. A password associated with the file. If you don't specify this argument, the file will have no password.

addtorecentfiles

(Optional) Variant (Boolean). Specifies whether to add the file to the most recent file list (value True) or not to add it (value False). The default is False.

Return values

None

{button ,AL(`H_123_SAVE_METHOD_MEMDEF;H_123_SAVECOPYAS_METHOD_MEMDEF;H_123_CLOSE_METHOD_MEMDEF;H_123_CLOSEALL_METHOD_MEMDEF;H_123_OPENDOCUMENT_METHOD_MEMDEF',0)} See related topics

1-2-3: SaveAsToInternet method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENDOCUMENTFROMINTERNET_METHOD_EXSCRIPT',1)} See example

Saves the document to a specified Universal Resource Locator (URL) on the Internet.

Syntax

document.**SaveAsToInternet** [*url*], [*filetype*], [*docpassword*], [*userid*], [*userpassword*], [*passiveconnection*], [*proxyserver*], [*proxyport*]

Parameters

url

(Optional) String. The URL for the document (for example,

"ftp://myftpserver/users/bob/test.123").

If you omit this argument, a dialog box prompts the user for it.

filetype

(Optional) String. The file format to be used. The following table lists the allowed values for this parameter.

Value	Description
"1-2-3 (123)"	1-2-3 97 workbook
"1-2-3 (WK4)"	1-2-3 for Windows Release 4
"1-2-3 (WK3)"	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
"1-2-3 (WK1)"	1-2-3 for DOS Release 2
"SmartMaster (12M)"	1-2-3 97 SmartMaster template
"Text (TXT)"	Text
"Excel Workbook (XLW)"	Microsoft Excel workbook
"Excel Worksheet (XLS)"	Microsoft Excel worksheet

docpassword

(Optional) String. A password associated with the document.

userid

(Optional) String. The user login name on the remote server. If you specify this argument, you must also supply the *userpassword* and *passiveconnection* arguments.

userpassword

(Optional) String. The user login password on the remote server.

passiveconnection

(Optional) Variant (Boolean). Specifies whether a passive connection to the remote server should be used (value True) or not (value False).

proxyserver

(Optional) String. The Internet proxy server IP address or domain name. If you specify this argument, you must also supply the *proxyport* argument.

proxyport

(Optional) Long. The port number to use to connect to the proxy server.

Return values

None

Usage

Although you can open a World Wide Web document using the OpenDocumentFromInternet method, you can't

directly save a document to the World Wide Web, so the proxy type always reflects saving the document to an FTP site via the File Transfer Protocol.

{button ,AL(`H_123_OPENDOCUMENTFROMINTERNET_METHOD_MEMDEF;H_123_RETRIEVEFILEFROMINTERNET_METHOD_MEMDEF;H_123_SETINTERNETOPTIONS_METHOD_MEMDEF',0)} See related topics

1-2-3: SaveAsToNotes method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_OPENDOCUMENTFROMNOTES_METHOD_EXSCRIPT',1)} See example

Saves the document as a Notes attachment.

Syntax

document.**SaveAsToNotes** [*attachedfilename*], [*universalnotesid*], [*fieldname*], [*databasefile*], [*servername*], [*filetype*], [*docpassword*]

Parameters

attachedfilename

(Optional) String. The name of the attached file in the Notes document. For example, "test.123". If you omit this argument, a dialog box prompts the user for it.

universalnotesid

(Optional) String. The 32-character hexadecimal Notes document ID (the NotesDocument.UniversalID property). For example, "150DFE45F1089B790065828D852562CA".

fieldname

(Optional) String. The name of the field in which the file is attached. For example, "Body".

databasefile

(Optional) String. The Notes database location. For example, "Databases\Docs_in_Progress.nsf".

servername

(Optional) String. The Notes server name. For example, "Local".

filetype

(Optional) String. The file format to be used. The following table lists the allowed values for this parameter.

Value	Description
"1-2-3 (123)"	1-2-3 97 workbook
"1-2-3 (WK4)"	1-2-3 Releases 4 and 5 for Windows
"1-2-3 (WK3)"	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
"1-2-3 (WK1)"	1-2-3 for DOS Release 2
"SmartMaster (12M)"	1-2-3 97 SmartMaster template
"Text (TXT)"	Text
"Excel Workbook (XLW)"	Microsoft Excel workbook
"Excel Worksheet (XLS)"	Microsoft Excel worksheet

docpassword

(Optional) String. A password associated with the 1-2-3 file.

Return values

None

Usage

When you save a document with the SaveAsToNotes method, the current document becomes the file copy you just attached to the Notes document, not the document you started with. The current document is now the same as if you had launched the attached file from the Notes document.

{button ,AL(`H_123_OPENDOCUMENTFROMNOTES_METHOD_MEMDEF',0)} See related topics

1-2-3: SaveCopyAs method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Saves a copy of the document in a specified file format, using a specified file name and path. The original document is not saved or changed.

Syntax

document.**SaveCopyAs** *name,* [*location*], [*filetype*], [*backup*], [*password*], [*comment*]

Parameters

name

String. The name of the file copy, or the name and full path of the file copy.

location

(Optional) String. The path where the file copy is to be saved. For example, "d:\lotus\work\123"

filetype

(Optional) String. The file format to be used. The default is "1-2-3 (123)" format. The following table lists the allowed values for this parameter.

Value	Description
"1-2-3 (123)"	1-2-3 97 workbook (default)
"1-2-3 (WK4)"	1-2-3 for Windows Releases 4, 5
"1-2-3 (WK3)"	1-2-3 for DOS Releases 3, 4; 1-2-3 for Windows Release 1
"1-2-3 (WK1)"	1-2-3 for DOS Release 2
"SmartMaster (12M)"	1-2-3 97 SmartMaster template
"Text (TXT)"	Text
"Excel Worksheet (XLS)"	Microsoft Excel worksheet
"Excel Workbook (XLW)"	Microsoft Excel workbook

backup

(Optional) Variant (Boolean). Specifies whether to back up the file (value True) or not (value False). If you specify False for this argument, 1-2-3 replaces any identically named and located file with the one you are saving.

password

(Optional) String. A password for the file. If you don't specify this argument, the file copy will have no password.

comment

(Optional) String. A description of the file contents. 1-2-3 places this string in the Document.Description property; it appears in the File - Open dialog box.

Return values

None

{button ,AL(`H_123_SAVE_METHOD_MEMDEF;H_123_SAVEAS_METHOD_MEMDEF;H_123_CLOSE_METHOD_MEMDEF;H_123_CLOSEALL_METHOD_MEMDEF;H_123_OPENDOCUMENT_METHOD_MEMDEF',0)} See related topics

1-2-3: ScrollToActiveCell method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_SCROLLTOACTIVECELL_METHOD_EXSCRIPT',1)} See example

Scrolls the current window to display the active cell.

Syntax

sheet.**ScrollToActiveCell**

Parameters

None

Return values

None

{button ,AL(`;H_123_PAGEBACK_METHOD_MEMDEF;H_123_PAGEFORWARD_METHOD_MEMDEF;H_123_TURNTO_METHOD_MEMDEF',0)} See related topics

1-2-3: Select method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_PLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERY_CLASS;H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_SELECT_METHOD_EXSCRIPT;H_123_SHOWVERSIONBORDERS_PROPERTY_EXSCRIPT;',1)} See example

Selects an object. Any objects in the same file that were previously selected become unselected.

Syntax

object.**Select**

Parameters

None

Return values

None

{button ,AL(`H_123_CLEARSELECTION_METHOD_MEMDEF;H_123_COPYSELECTION_METHOD_MEMDEF;H_123_CUTSELECTION_METHOD_MEMDEF;H_123_EXTENDSELECTFROMTAB_METHOD_MEMDEF;H_123_EXTENDSELECT_METHOD_MEMDEF;H_123_EXTENDWORKSHEETSELECTIONBACK_METHOD_MEMDEF;H_123_EXTENDWORKSHEETSELECTIONFORWARD_METHOD_MEMDEF;H_123_GETSELECTION_METHOD_MEMDEF;H_123_REMOVEFROMSELECTION_METHOD_MEMDEF;H_123_SELECT_METHOD_MEMDEF;H_123_SELECTALL_METHOD_MEMDEF;H_123_SELECTALLSHEETS_METHOD_MEMDEF;H_123_ACTIVE_PROPERTY_MEMDEF;H_123_ISSELECTABLE_PROPERTY_MEMDEF;H_123_ISSELECTED_PROPERTY_MEMDEF;H_123_SELECTION_PROPERTY_MEMDEF;H_123_SELECTED_EVENT_MEMDEF',0)} See related topics

1-2-3: SelectAll method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_SELECT_METHOD_EXSCRIPT',1)} See example

Selects the active area of the current sheet.

Syntax

sheet.**SelectAll**

Parameters

None

Return values

None

Usage

The active area of a sheet is the area bounded by cell A1 and the lowest and rightmost nonblank cell in the current sheet.

1-2-3: SelectAllSheets method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

Selects the active areas of the all sheets in the current file.

Syntax

sheet.**SelectAllSheets**

Parameters

None

Return values

None

Usage

The active area of a sheet is the area bounded by cell A1 and the lowest and rightmost nonblank cell in the current sheet.

1-2-3: Send method

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_SEND_METHOD_EXSCRIPT',1)} See example

Sends the workbook or range in an e-mail message, using the local e-mail application, while the user is in 1-2-3.

Syntax

object.**Send** [*recipients*], [*individualmessages*], [*subject*], [*body*], [*returnreceipt*], [*priority*], [*extractformulas*], [*sendtype*], [*returntooriginator*], [*messagetooriginator*], [*messagetoalternate*], [*messagewithtrack*], [*modifyroute*], [*savecopy*]

Parameters

recipients

(Optional) String. The list of mail recipients (with names separated by semicolons).

individualmessages

(Optional) String. The list of individual messages to recipients (with messages separated by semicolons).

subject

(Optional) String. The subject of the e-mail message.

body

(Optional) String. The text of the e-mail message.

returnreceipt

(Optional) Variant (Boolean). Determines whether to send a delivery confirmation or not.

Value	Description
True	Send a confirmation to each sender when a recipient opens the e-mail message.
False	Don't send a confirmation (default).

priority

(Optional) Variant (MailPriority enumeration). The delivery priority. The following table lists the allowed values for this parameter.

Value	Description
$LowPriority	Low priority
$NormalPriority	Normal priority (default)
$HighPriority	High priority

extractformulas

(Optional) Variant (WALExtractOption enumeration). Determines whether to extract formulas from the cells. The following table lists the allowed values for this parameter. This parameter only has meaning if the object you run this method on is a range.

Value	Description
$ExtractValuesAndFormulas	Extract formulas and values (default).
$ExtractValuesOnly	Extract values only.

sendtype

(Optional) Variant (SendType enumeration). Determines how to distribute the message to the recipients. The following table lists the allowed values for this parameter.

Value	Description
$Route	Route the message sequentially. Each recipient must forward it to the next (default when you run this method on a

	Range object).
$Broadcast	Send the message to all recipients at the same time (default when you run this method on a Document object).

returntooriginator

(Optional) Variant (Boolean). Determines whether to return the message to the originator when all recipients have received the message.

Value	Description
True	Return the message to the originator after all recipients have received the message (default when you run this method on a Range object).
False	Don't return the message to the originator after all recipients have received the message (default when you run this method on a Document object).

messagetooriginator

(Optional) Variant (Boolean). Determines whether to send a tracking message to the originator each time the message is forwarded to the next recipient (value True, the default) or not (value False). This parameter only has meaning if *sendtype* is $Route.

alternaterecipient

(Optional) String. The name of an alternate person to receive a tracking message when a recipient forwards the message. This parameter only has meaning if *sendtype* is $Route.

messagewithtrack

(Optional) Variant (Boolean). Determines whether to attach the routed workbook to tracking messages (value True) or not (value False, the default). This parameter only has meaning if *sendtype* is $Route and if *messagetooriginator* is True or an *alternaterecipient* is specified.

modifyroute

(Optional) Variant (Boolean). Determines whether recipients are allowed to alter the routing list of later recipients (value True) or not (value False, the default). This parameter only has meaning if *sendtype* is $Route.

savecopy

(Optional) Variant (Boolean). Determines whether to save a copy of the message (value True, the default) or not (value False).

Return values

None

Usage

Call the UserLogin method before calling the Send method, to prevent the display of the user mail login dialog box.

If you don't specify any valid recipients in the *recipients* argument, the SendMail method brings up a dialog box for the user to specify the list of recipients.

To pass a list of strings in an argument, use the data type String and delimit each string in the list with a semicolon.

The Send method cannot be recorded.

{button ,AL(`H_123_SENDMAIL_METHOD_MEMDEF;H_123_USERLOGIN_METHOD_MEMDEF',0)} See related topics

1-2-3: SendCommand method

{button ,AL(`H_123_SENDCOMMAND_METHOD_MEMDEF_RT;H_123_DOCUMENT_CLASS;',0)} See list of classes

Connects to the specified database and sends a database command.

Syntax

document.**SendCommand** *drivername,* [*driveruserid,*] [*driverpassword,*] [*connectstring,*] *databasename,* [*databaseuserid,*] [*databasepassword,*] *commandstring*

Parameters

driverName

String. The name of the driver to use for the connection, for example, "dBase_IV."

driverUserID

(Optional) String. The database driver user ID.

driverpassword

(Optional) String. The database driver password.

connectString

(Optional) String. A connection string to pass to the driver. Use this to specify additional information that may be needed to connect to the database.

databaseName

String. The path and name of the external database to which you want to connect, for example, "C:\LOTUS\APPROACH\STAFF.DBF."

databaseUserID

(Optional) String. The database user ID.

databasepassword

(Optional) String. The database password.

commandstring

String. The string containing the database commands to send to the specified database.

Return values

None

{button ,AL(`;H_123_SEND_METHOD_MEMDEF;H_123_SENDSQL_METHOD_MEMDEF;',0)} See related topics

1-2-3: SendMail method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_SENDMAIL_METHOD_EXSCRIPT',1)} See example

Sends an e-mail message, using the local e-mail application, while the user is in 1-2-3. The e-mail can contain text, the current selection, or both.

Syntax

application.**SendMail** [*recipients*], [*individualmessages*], [*subject*], [*body*], [*returnreceipt*], [*priority*], [*attachment*], [*savecopy*]

Parameters

recipients

(Optional) String. The list of e-mail recipients, with each name separated by a ; (semicolon).

individualmessages

(Optional) String. The list of individual messages to recipients, with each message separated by a ; (semicolon).

subject

(Optional) String. The subject of the e-mail message.

body

(Optional) String. The text of the e-mail message.

returnreceipt

(Optional) Variant (Boolean). Determines whether to send a delivery confirmation or not. This parameter has no effect if the *attachment* parameter is $SelectedObject or $Clipboard.

Value	Description
True	Send a confirmation to each sender when a recipient opens the mail message.
False	Don't send a confirmation (default).

priority

(Optional) Variant (MailPriority enumeration). The delivery priority. This parameter has no effect if the *attachment* parameter is $SelectedObject or $Clipboard. The following table lists the allowed values for this parameter.

Value	Description
$LowPriority	Low priority
$NormalPriority	Normal priority (default)
$HighPriority	High priority

attachment

(Optional) Variant (Attachment enumeration). Specifies an attachment to send with the message. The following table lists the allowed values for this parameter.

Value	Description
$SelectedObject	Send the currently selected object.
$Clipboard	Send the contents of the Clipboard via DDE.
$NoAttachment	Don't send an attachment (default).

savecopy

(Optional) Variant (Boolean). Determines whether to save a copy of the message (value True, the default) or not to save it (value False).

Return values
None

Usage
Call the UserLogin method before calling the SendMail method, to prevent the display of the user mail login dialog box when the user's mail application is not running.

If you don't specify any valid recipients in the *recipients* argument, the SendMail method brings up a dialog box for the user to specify the list of recipients.

To pass a list of strings in an argument, use the data type String and delimit each string in the list with a ; (semicolon).

The SendMail method cannot be recorded.

{button ,AL(`H_123_SEND_METHOD_MEMDEF;H_123_USERLOGIN_METHOD_MEMDEF',0)} See related topics

1-2-3: SendSQL method

{button ,AL(`H_123_SENDSQL_METHOD_MEMDEF_RT;H_123_DOCUMENT_CLASS;',0)} See list of classes

Sends a Structured Query Language (SQL) statement to the specified database. If the database cannot return results from the SQL statement, no results are returned.

Syntax

document.**SendSQL** *externalrangename, commandstring,* [*outputrange,*] [*errorcodelocation*]

Parameters

externalrangename

Variant. The name of the external range representing an existing connection to an external database.

commandstring

String. The SQL statement sent to the external database.

outputrange

(Optional) Variant. The name of the range in which to place the output that is sent from the SQL statement to the external database.

errorcodelocation

(Optional) Variant. The name of the range to contain any error output from the SQL statements sent to the external database.

Return values

None

{button ,AL(`;H_123_SEND_METHOD_MEMDEF;H_123_SENDCOMMAND_METHOD_MEMDEF;',0)} See related topics

1-2-3: SetActiveCell method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_SETACTIVECELL_METHOD_EXSCRIPT;',1)} See example

Makes the top left cell in the selected range the active cell.

Syntax

range.**SetActiveCell** ([*rangemember*])

Parameters

rangemember

(Optional) Variant. The range in which to position the active cell, if the selection contains more than one range. The default is to make the top left cell in the last range added to the collection of ranges the active cell.

Return values

None

1-2-3: SetCellData method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_FREECELLDATA_METHOD_EXSCRIPT;H_123_SETCELLDATA_METHOD_EXSCRIPT',1)} See example

Fills a range with data passed from an external C routine.

Syntax

range.**SetCellData** *pointer*

Parameters

pointer

Long. A pointer to the data you want to enter in the range. This data includes a header and an array of cell data pointers.

Return values

None

{button ,AL(`H_123_GETCELLDATA_METHOD_MEMDEF;H_123_FREECELLDATA_METHOD_MEMDEF',0)} See related topics

1-2-3: SetGalleryStyle method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_SETGALLERYSTYLE_METHOD_EXSCRIPT ',1)} See example

Applies a style template from the Style Gallery to the specified range.

Syntax

range.**SetGalleryStyle** *gallerystylename*

Parameters

gallerystylename

Variant (GalleryStyles enumeration). The name of a style template to apply to the specified range. The following table lists the allowed values for this parameter.

Value	Description
$Chisel1	Style with chiseled borders.
$Chisel2	Style with chiseled borders.
$Computer	Style with green and white bands resembling a computer printout.
$Photo	Style with frame corners and borders resembling a photo album.
$Picture1	Style with borders resembling a desktop photo.
$Picture2	Style with purple and grey regions.
$NotePad	Style with a yellow background and dog-eared corner resembling a post-it note.
$Simple1	Style with purple lines and text.
$Simple2	Style with green regions and text.
$Simple3	Style with blue regions and text.
$B&W1	Style with horizontal rules for each row.
$B&W2	Style with a grey region for the first row and horizontal rules for each row.
$B&W3	Style with horizontal rules for each row.
$B&W4	Style with minimal rules and character accents.

Return values

None

{button ,AL(`;H_123_DEFINENAMEDSTYLE_METHOD_MEMDEF;H_123_MODIFYNAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTONAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTOSTYLE_METHOD_MEMDEF',0)} See related topics

1-2-3: SetHorizontalTitle method

{button ,AL(`H_123_SETHORIZTITLE_METHOD_MEMDEF_RT;H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_SETHORIZONTALTITLE_METHOD_EXSCRIPT ',1)} See example

Sets the coordinates for the title along the top edge of a sheet.

Syntax

sheet.**SetHorizontalTitle** *titlerange, numberofrows*

Parameters

titlerange

Variant. The name of the range specifying the starting position for the horizontal title.

numberofrows

Long. The number of rows to include in the horizontal title.

Return value

None

{button ,AL(`;H_123_SETVERTTITLE_METHOD_MEMDEF',0)} See related topics

1-2-3: SetInternetOptions method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_SETINTERNETOPTIONS_METHOD_EXSCRIPT',1)} See example

Displays the File - Internet - FTP Connection Setup dialog box, in which the user can set Internet options.

Syntax

application.**SetInternetOptions**

Parameters

None

Return values

None

{button ,AL(`H_123_OPENDOCUMENTFROMINTERNET_METHOD_MEMDEF;H_123_SAVEASTOINTERNET_METHOD_MEMDEF;H_123_RETRIEVEFILEFROMINTERNET_METHOD_MEMDEF',0)} See related topics

1-2-3: SetOrigin method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_SETORIGIN_METHOD_EXSCRIPT ',1)} See example

Scrolls to display the top-left cell of a specified range in the top-left corner of the display region. The current selection does not change.

Syntax

sheet.**SetOrigin** *originposition*

Parameters

originposition

Variant. The address or range name whose top-left cell serves as the new top-left cell displayed in the current window.

Return values

None

{button ,AL(`;H_123_MOVE_METHOD_MEMDEF;H_123_MOVEORIGIN_METHOD_MEMDEF',0)} See related topics

1-2-3: SetRecordsLimitMax method

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_SORTDATA_METHOD_EXSCRIPT;',1)} See example

Sets a maximum number of records to be retrieved from a query and specifies whether that limit is applied to the current query.

Syntax

query.**SetRecordsLimitMax** *limitrecords,* [*numberofrecords*]

Parameters

limitrecords

Boolean (enumeration). Specifies whether the query should or should not retrieve a maximum number of records for the current query. The following table lists the allowed values for this parameter.

Value	Description
Yes	The current query should retrieve no more than the number of records specified in the *numberofrecords* parameter.
No	The current query can retrieve an unrestricted number of records.

numberofrecords

(Optional) Long. The maximum number of records that the query can retrieve if the value of *limitrecords* is Yes. The range of valid values for *numberofrecords* is any long from 1 - 8191 (the maximum number of rows in a sheet).

Return values

None

{button ,AL(`;H_123_JOIN_METHOD_MEMDEF;H_123_REFRESH_METHOD_MEMDEF;H_123_UPDATE_METHOD_MEMDEF',0)} See related topics

1-2-3: SetVerticalTitle method

{button ,AL(`H_123_SETVERTTITLE_METHOD_MEMDEF_RT;H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_SETVERTICALTITLE_METHOD_EXSCRIPT',1)} See example

Sets the coordinates for the title along the left edge of a sheet.

Syntax

sheet.**SetVerticalTitle** *titlerange, numberofcolumns*

Parameters

titlerange

Variant. The name of the range specifying the starting position for the vertical title.

numberofcolumns

Long. The number of columns to include in the vertical title.

Return value

None

{button ,AL(`H_123_SETHORIZTITLE_METHOD_MEMDEF',0)} See related topics

1-2-3: Show method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_SHOW_METHOD_EXSCRIPT ',1)} See example

Displays a Workbook window for the specified file, which has been opened invisibly.

Syntax

document.**Show**

Parameters

None

Return values

None

{button ,AL(`H_123_SHOWALLSHEETS_METHOD_MEMDEF',0)} See related topics

1-2-3: ShowAllSheets method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_SHOWALLSHEETS_METHOD_EXSCRIPT ',1)} See example

Shows all hidden sheets in a file.

Syntax

document.**ShowAllSheets**

Parameters

None

Return values

None

{button ,AL(`H_123_SHOW_METHOD_MEMDEF',0)} See related topics

1-2-3: ShowIconBar method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Displays the specified icon bar (set of SmartIcons).

Syntax

applicationwindow.**ShowIconBar** *iconbarname*

Parameters

iconbarname

String. The name of the icon bar to be shown.

Return values

None

Usage

The ApplicationWindow.IconBarNames property is a collection of names of known icon bars that you can show or hide.

{button ,AL(`H_123_HIDEICONBAR_METHOD_MEMDEF;H_123_ICONBARNAMES_PROPERTY_MEMDEF',0)} See related topics

1-2-3: ShowSheet method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_SHOWSHEET_METHOD_EXSCRIPT ',1)} See example

Displays the specified sheet.

Syntax

sheet.**ShowSheet**

Parameters

None

Return values

None

Usage

Use the ShowSheet method to display sheets that have been hidden using the HideSheet method.

{button ,AL(`H_123_SHOW_METHOD_MEMDEF;H_123_SHOWALLSHEETS_METHOD_MEMDEF;H_123_HIDESHEETS_METHOD_MEMDEF;',0)} See related topics

1-2-3: SmartSum method

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_SMARTSUM_METHOD_EXSCRIPT',1)} See example

Sums values in a range or adjacent range, if you include empty cells below or to the right of the range.

Syntax

range.**SmartSum**

Parameters

None

Return values

None

1-2-3: Sort method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RANGESORTDEFINEKEY_METHOD_EXSCRIPT',1)} See example

Sorts data in a range, using the sort keys defined by the RangeSortDefineKey method.

Syntax

range.**Sort**

Parameters

None

Return values

None

1-2-3: SortData method

{button ,AL(`H_SORTDATA_METHOD_MEMDEF_RT;H_123_QUERY_CLASS;',0)} See list of classes

{button ,AL(`H_123_SORTDATA_METHOD_EXSCRIPT;',1)} See example

Sorts output records from the current query without reexecuting the query.

Syntax

query.**SortData** [*fieldname1, sortdirection1, fieldname2, sortdirection2, fieldname3, sortdirection3*]

Parameters

fieldname1

(Optional) String. The name of the field to be used as the first column to sort by. For example "LastName."

fieldname2

(Optional) String. The name of the field to be used as the second column to sort by. For example "FirstName."

fieldname3

(Optional) String. The name of the field to be used as the third column to sort by. For example "HomeTown."

sortdirection1, sortdirection2, sortdirection3

(Optional) Variant (Direction enumeration). The direction of the sort for the first, second, and third columns to sort by, respectively.The following table lists the allowed values for this parameter.

Value	Description
$Ascend	Sorts from A - Z or from 0 (zero) to the highest number.
$Descend	Sorts from Z - A or from the highest number to 0 (zero).

Return values

None

{button ,AL(`H_123_QUERYSORTDEFINEKEY_METHOD_MEMDEF;H_123_SORT_METHOD_MEMDEF;',0)} See related topics

1-2-3: SortReset method

{button ,AL(`H_SORTRESET_METHOD_MEMDEF_RT;H_123_QUERY_CLASS;',0)} See list of classes

{button ,AL(`H_123_SORTDATA_METHOD_EXSCRIPT;',1)} See example

Resets all the columns to sort by on all fields for the current query.

Syntax

query.**SortReset**

Parameters

None

Return values

None

{button ,AL(`H_123_QUERYSORTDEFINEKEY_METHOD_MEMDEF;H_123_SORT_METHOD_MEMDEF;H_123_SORTDATA_METHOD_MEMDEF;',0)} See related topics

1-2-3: SortResetKeys method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_RANGESORTDEFINEKEY_METHOD_EXSCRIPT',1)} See example

Clears all sort keys for sorting range data.

Syntax

range.**SortResetKeys**

Parameters

None

Return values

None

1-2-3: StartPoll method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTPOLL_METHOD_EXSCRIPT',1)} See example

Starts the specified poll. You can activate four separate polls, at different event time intervals and raising different numbers of events.

Syntax

document.**StartPoll** *eventindex, timeinterval, repetitions*

Parameters

eventindex

Long. The index of the poll event to start. There are four different poll events available, which can be operating concurrently. The value *eventindex* = 1 specifies the event Poll1, and so on through the event Poll4.

timeinterval

Long. The time interval between successive poll events, in milliseconds. Higher-priority system tasks can lengthen a particular event interval.

repetitions

Long. The number of times to raise this event. The value zero causes the events to go on indefinitely, until you call the EndPoll method to turn off the poll, or close the document.

Return values

None

{button ,AL(`H_123_ENDPOLL_METHOD_MEMDEF;H_123_POLL1_EVENT_MEMDEF;H_123_POLL2_EVENT_MEMDEF;H_123_POLL3_EVENT_MEMDEF;H_123_POLL4_EVENT_MEMDEF',0)} See related topics

1-2-3: StyleFontReset

{button ,AL(`H_123_RANGE_Class;',0)} See list of classes

{button ,AL(`H_123_STYLEFONTRESET_METHOD_EXSCRIPT',1)} See example

Restores default font, point size, attribute, and color for labels and values in a range.

Syntax

range.**StyleFontReset**

Parameters

None

Return values

None

{button ,AL(`;H_123_MODIFYNAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTONAMEDSTYLE_METHOD_MEMDEF;H_123_REVERTTOSTYLE_METHOD_MEMDEF',0)} See related topics

```
' Example: NewApproachConnection method
' Create a new ApproachConnection object.
Dim approachconnection As ApproachConnection
Set approachconnection = [A].NewApproachConnection(4100, 1100, 4100, 1100, "Approach
Report",,, True,,,, [A:B2..A:F30])
```

```
' Example: NewArc method
' Create a 1" x 1" arc on sheet A.
Dim arc1 As Arc
Set arc1 = [A].NewArc(1440, 1440, 2880, 2880)
```

```
' Example: NewChart method
' Create a new chart of the data in [A:B2..A:D10].
' The chart's object reference variable is not needed.
[A:B2].Contents = "10"
[A:B3].Contents = "20"
[A:B4].Contents = "30"
[A].NewChart 1440, 1440, 7200, 7200, [A:B2..A:D10]
```

```
' Example: NewDataLink method
' To run this example, substitute the name of a WordPro document and bookmark
' for the filename (LnkTst.lwp) and itemname (test1) in this example.

' Create a new data link at [G1].
Dim datLnk1 As DataLink
Set datLnk1 = CurrentDocument.NewDataLink("Datalink1", _
                              "E:\LnkTst.lwp", "test1", $TextFormat, True)
Set datLnk1.Target = [G1]
```

```
' Example: NewMap method
' Create a data map of the U.S.A.
' The color of each state on the map indicates its corresponding cell value.
[A:B2].Contents = "CA"
[A:B3].Contents = "MA"
[A:B4].Contents = "NY"
[A:C2].Contents = "20"
[A:C3].Contents = "30"
[A:C4].Contents = "40"
[A].NewMap 3320, 3100, 8510, 8000, [A:B2..A:C4], "USA by State"
```

```
' Example: NewObject method
' Create a new Paint picture.
[A].NewObject 1440, 1440, 4320, 4320, "Paint.Picture",,,,,,
```

```
' Example: NewQueryTable method
' Create a new QueryTable object.
Dim queryTable1 As QueryTable
' Enter field names for the input range.
[A:B2].Contents = "Field 1"
[A:C2].Contents = "Field 2"
[A:D2].Contents = "Field 3"
' Enter data for a record.
[A:B3].Contents = "1"
[A:C3].Contents = "2"
[A:D3].Contents = "3"
' Create the query table.
Set queryTable1 = [A].NewQueryTable(2880, 2880, 7200, 7200, _
                "ApproachWorksheet",,,,,,, [A:B2..A:D5], [A:F2..A:H5])
```

```
' Example: NewSheet method
' Insert 3 new sheets before the current sheet.
Dim sheet1 As Sheet
Set sheet1 = CurrentDocument.NewSheet($Before, 3, True)
sheet1.Name = "Sheet 1"
```

```
' Example: NewVersion method
' Create a new version for a range.
CurrentDocument.CreateRangeName "Range_1", [A:B2..A:B10]
[Range_1].NewVersion "Version_2"
' Do something with the version ...
' If necessary, delete the version.
[Range_1.Version_2].DeleteVersion
```

```
' Example: Open and Next methods
' Open a range collection, retrieve each of its items, and add a new version to each.
Sub processRanges
   Dim collection As Ranges         ' Collection to extract items from
   Dim item As Variant              ' Item retrieved from collection
   Dim itemIndex As Integer         ' Loop index for collection items
   ' Add some items to the range collection.
   CurrentDocument.CreateRangeName "Range 1", [A:B2..A:B10]
   CurrentDocument.CreateRangeName "Range 2", [A:C2..A:C10]
   ' Retrieve all the items in the range collection and process them.
   Set collection = CurrentDocument.Ranges
   If collection.Count <> 0 Then
      Set item = collection.Open
      If IsEmpty(item) = False Then Call myItemProcessor(item)
   End If
   For itemIndex = 2 To collection.Count
      Set item = collection.Next
      If IsEmpty(item) = False Then Call myItemProcessor(item)
   Next
End Sub

' Process each item retrieved. For example, create a new version.
Sub myItemProcessor(item As Variant)
   item.NewVersion "Version X"
   item.VersionBorderVisible = True
End Sub
```

```
' Example: ViewSplitStyle property; NextSplit and ClearSplits methods
' Split the current document horizontally, move between panes,
' and then clear the splits.
CurrentDocument.ViewSplitStyle = $Horizontal
CurrentDocument.NextSplit
MessageBox "Clearing splits.", MB_OK + MB_ICONINFORMATION
CurrentDocument.ClearSplits
```

```
' Example: Sheets property; Close, OpenDocument, and SaveAs methods
' Save a 1-2-3 workbook file, close it, and then open it as read-write and visible.

' First create a test document.
Dim document1 As Document
' Open a new document without specifying a file or path.
Set document1 = CurrentApplication.NewDocument
' Add some content to the document.
' For example, put some text into the first sheet name.
document1.Sheets(0).Name = "My sheet 1"
' Save to a file.
document1.SaveAs "testopen.123", "d:\lotus\work\123", "1-2-3 (123)", False,
"mypassword", True
' Add more content to the document.
' For example, put some text into the first column head.
[A:A1].Contents = "My Column Head 1"
' Close the document, saving the changes.
document1.Close True
' Later, open the file.
Set document1 = CurrentApplication.OpenDocument("testopen.123", _
                "d:\lotus\work\123", "1-2-3 (123)", "mypassword", False, True, True,,)
```

```
' Example: OpenDocumentFromInternet and SaveAsToInternet methods
' Save a test file to the Internet and then open it from the Internet.
Dim file1 As Document
Set file1 = .NewDocument
[A:A1].Contents = "Internet File 1"
' Save file to FTP URL name without proxy.
file1.SaveAsToInternet "ftp://myintranet/lotus/file1.123", "1-2-3 (123)", "",
"userid", "userpassword"
' Close the file.
file1.Close
' Later, open the file by FTP URL name without proxy.
.OpenDocumentFromInternet "ftp://myintranet/lotus/file1.123", "1-2-3 (123)",,,
"userid", "userpassword",,,
```

```
' Example: OpenDocumentFromNotes and SaveAsToNotes methods
' Save a test file to a Notes database and then open it from Notes.

' First create a test file.
Dim file1 As Document
Set file1 = .NewDocument
[A:A1].Contents = "Notes File 1"

' Save the file to a local Notes database.
.SaveAsToNotes "file1.123", "0123456789ABCDEF0123456789ABCDEF", "Body", _
               "MyNotesDB.nsf", "Local", "1-2-3 (123)"

' Open the file from Notes.
.OpenDocumentFromNotes "file1.123", "0123456789ABCDEF0123456789ABCDEF", _
                       "Body", "MyNotesDB.nsf", "Local", _
                       "1-2-3 (123)",, True

' A script you can use in Notes Version 4 to extract a document's universal ID.
     Dim workspace As New NotesUIWorkspace
     Dim uidoc As NotesUIDocument
     Dim doc As NotesDocument
     Dim unid As String

     ' Get the universal ID.
     Set uidoc = workspace.CurrentDocument
     Set doc = uidoc.Document
     unid = doc.UniversalID

     ' Insert the universal ID into the Body field of the document.
     uidoc.EditMode = True
     uidoc.GotoField("Body")
     uidoc.InsertText(unid)
```

```
' Example: OutlineColumnsToLevel, OutlineRowsToLevel,
'          PromoteColumn, and PromoteRow methods
Sub OutlineDemo
   Msgbox "Create outline levels for columns B - E."
   ' Demote columns B, C, D, and E by one outline level.
   [A:B1..A:E8192].DemoteColumn 1
   ' Demote columns C, D, and E by one outline level.
   [A:C1..A:E8192].DemoteColumn 1

   ' Promote column D by one outline level.
   Msgbox " Promote column D by one outline level."
   [A:D1..A:D8192].PromoteColumn 1

   Msgbox "Create outline levels for rows 2 - 5."
   ' Demote rows 2, 3, 4, and 5 by one outline level.
   [A:A2..A:IV5].DemoteRow 1
   ' Demote rows 3, 4, and 5 by one outline level.
   [A:A3..A:IV5].DemoteRow 1
   ' Promote row 4 by one outline level.
   Msgbox " Promote row 4 by one outline level."
   [A:A4..A:IV4].PromoteRow 1

   ' Collapse columns and rows to display outline at levels 0 - 1.
   Msgbox "Collapse columns and rows to display outline at levels 0 - 1."
   [A].OutlineColumnsToLevel 1
   [A].OutlineRowsToLevel 1

   ' Collapse columns and rows to display the outline at level 0.
   Msgbox "Collapse columns and rows to display the outline at level 0."
   [A].OutlineColumnsToLevel 0
   [A].OutlineRowsToLevel 0

   ' Expand columns and rows to display the outline at levels 0 - 2.
   Msgbox "Expand columns and rows to display the outline at levels 0 - 2."
   [A].OutlineColumnsToLevel 2
   [A].OutlineRowsToLevel 2

   ' Clear all outline levels for the current sheet.
   Msgbox "Clear the outline."
   [A].ClearOutline

End Sub
```

```
' Example: OutlineColumnsToLevel, OutlineRowsToLevel,
'          PromoteColumn, and PromoteRow methods
Sub OutlineDemo
   Msgbox "Create outline levels for columns B - E."
   ' Demote columns B, C, D, and E by one outline level.
   [A:B1..A:E8192].DemoteColumn 1
   ' Demote columns C, D, and E by one outline level.
   [A:C1..A:E8192].DemoteColumn 1

   ' Promote column D by one outline level.
   Msgbox " Promote column D by one outline level."
   [A:D1..A:D8192].PromoteColumn 1

   Msgbox "Create outline levels for rows 2 - 5."
   ' Demote rows 2, 3, 4, and 5 by one outline level.
   [A:A2..A:IV5].DemoteRow 1
   ' Demote rows 3, 4, and 5 by one outline level.
   [A:A3..A:IV5].DemoteRow 1
   ' Promote row 4 by one outline level.
   Msgbox " Promote row 4 by one outline level."
   [A:A4..A:IV4].PromoteRow 1

   ' Collapse columns and rows to display outline at levels 0 - 1.
   Msgbox "Collapse columns and rows to display outline at levels 0 - 1."
   [A].OutlineColumnsToLevel 1
   [A].OutlineRowsToLevel 1

   ' Collapse columns and rows to display the outline at level 0.
   Msgbox "Collapse columns and rows to display the outline at level 0."
   [A].OutlineColumnsToLevel 0
   [A].OutlineRowsToLevel 0

   ' Expand columns and rows to display the outline at levels 0 - 2.
   Msgbox "Expand columns and rows to display the outline at levels 0 - 2."
   [A].OutlineColumnsToLevel 2
   [A].OutlineRowsToLevel 2

   ' Clear all outline levels for the current sheet.
   Msgbox "Clear the outline."
   [A].ClearOutline
End Sub
```

```
' Example: PageBack, PageForward, ShowAllSheets,
'          ShowSheet, and TurnTo methods
Sub NavigateDoc
   ' Create 10 new sheets after the current sheet in the workbook.
   .NewSheet $After, 10, True
   .PageBack 2
   ' Move forward three sheets in the workbook.
   Msgbox "Move forward three sheets."
   .PageForward 3
   ' Move back one sheet in the workbook.
   Msgbox "Move back one sheet."
   .PageBack

   ' Turn to sheet A in the workbook.
   Msgbox "Turn to sheet A in the workbook."
   [A].TurnTo
   ' Hide sheets B, C, and D.
   Msgbox "Hide sheets B, C, and D."
   [B].HideSheet
   [C].HideSheet
   [D].HideSheet

   ' Unhide sheet B.
   Msgbox "Unhide sheet B."
   [B].ShowSheet

   ' Unhide all hidden sheets (including sheets C and D).
   Msgbox "Unhide all hidden sheets (including sheets C and D)."
   .ShowAllSheets
End Sub
```

```
' Example: PageBack, PageForward, ShowAllSheets,
'          ShowSheet, and TurnTo methods
Sub NavigateDoc
   ' Create 10 new sheets after the current sheet in the workbook.
   .NewSheet $After, 10, True
   .PageBack 2
   ' Move forward three sheets in the workbook.
   Msgbox "Move forward three sheets."
   .PageForward 3
   ' Move back one sheet in the workbook.
   Msgbox "Move back one sheet."
   .PageBack

   ' Turn to sheet A in the workbook.
   Msgbox "Turn to sheet A in the workbook."
   [A].TurnTo
   ' Hide sheets B, C, and D.
   Msgbox "Hide sheets B, C, and D."
   [B].HideSheet
   [C].HideSheet
   [D].HideSheet

   ' Unhide sheet B.
   Msgbox "Unhide sheet B."
   [B].ShowSheet

   ' Unhide all hidden sheets (including sheets C and D).
   Msgbox "Unhide all hidden sheets (including sheets C and D)."
   .ShowAllSheets
End Sub
```

```
' Example: PrintSelection and PrintWhat properties; Preview method
' Display the print Preview window for a document.
' First, add some content to the current document.
[A:A1].Contents = "My First Heading"
' Select a range to preview.
[A:A1..A:C10].Select
Set CurrentDocument.CurrentPrintSettings.PrintSelection = [A:A1..A:C10]
CurrentDocument.CurrentPrintSettings.PrintWhat = $CurrentSelection
' Let the user change the print properties.
CurrentApplication.Preview
' Pause before proceeding to allow user to make changes in the Preview window.
' For example, yield control to the Preview & Page Setup InfoBox using the Yield
statement,
' put up a modeless dialog box using the operating system API,
' or terminate the LotusScript module.
```

```
' Example: Print method
' Print the current sheet.
CurrentDocument.CurrentPrintSettings.PrintWhat = $CurrentSheet
CurrentApplication.Print
```

```
' Example: OutlineColumnsToLevel, OutlineRowsToLevel,
'          PromoteColumn, and PromoteRow methods
Sub OutlineDemo
   Msgbox "Create outline levels for columns B - E."
   ' Demote columns B, C, D, and E by one outline level.
   [A:B1..A:E8192].DemoteColumn 1
   ' Demote columns C, D, and E by one outline level.
   [A:C1..A:E8192].DemoteColumn 1

   ' Promote column D by one outline level.
   Msgbox " Promote column D by one outline level."
   [A:D1..A:D8192].PromoteColumn 1

   Msgbox "Create outline levels for rows 2 - 5."
   ' Demote rows 2, 3, 4, and 5 by one outline level.
   [A:A2..A:IV5].DemoteRow 1
   ' Demote rows 3, 4, and 5 by one outline level.
   [A:A3..A:IV5].DemoteRow 1
   ' Promote row 4 by one outline level.
   Msgbox " Promote row 4 by one outline level."
   [A:A4..A:IV4].PromoteRow 1

   ' Collapse columns and rows to display outline at levels 0 - 1.
   Msgbox "Collapse columns and rows to display outline at levels 0 - 1."
   [A].OutlineColumnsToLevel 1
   [A].OutlineRowsToLevel 1

   ' Collapse columns and rows to display the outline at level 0.
   Msgbox "Collapse columns and rows to display the outline at level 0."
   [A].OutlineColumnsToLevel 0
   [A].OutlineRowsToLevel 0

   ' Expand columns and rows to display the outline at levels 0 - 2.
   Msgbox "Expand columns and rows to display the outline at levels 0 - 2."
   [A].OutlineColumnsToLevel 2
   [A].OutlineRowsToLevel 2

   ' Clear all outline levels for the current sheet.
   Msgbox "Clear the outline."
   [A].ClearOutline
End Sub
```

```
' Example: OutlineColumnsToLevel, OutlineRowsToLevel,
'          PromoteColumn, and PromoteRow methods
Sub OutlineDemo
   Msgbox "Create outline levels for columns B - E."
   ' Demote columns B, C, D, and E by one outline level.
   [A:B1..A:E8192].DemoteColumn 1
   ' Demote columns C, D, and E by one outline level.
   [A:C1..A:E8192].DemoteColumn 1

   ' Promote column D by one outline level.
   Msgbox " Promote column D by one outline level."
   [A:D1..A:D8192].PromoteColumn 1

   Msgbox "Create outline levels for rows 2 - 5."
   ' Demote rows 2, 3, 4, and 5 by one outline level.
   [A:A2..A:IV5].DemoteRow 1
   ' Demote rows 3, 4, and 5 by one outline level.
   [A:A3..A:IV5].DemoteRow 1
   ' Promote row 4 by one outline level.
   Msgbox " Promote row 4 by one outline level."
   [A:A4..A:IV4].PromoteRow 1

   ' Collapse columns and rows to display outline at levels 0 - 1.
   Msgbox "Collapse columns and rows to display outline at levels 0 - 1."
   [A].OutlineColumnsToLevel 1
   [A].OutlineRowsToLevel 1

   ' Collapse columns and rows to display the outline at level 0.
   Msgbox "Collapse columns and rows to display the outline at level 0."
   [A].OutlineColumnsToLevel 0
   [A].OutlineRowsToLevel 0

   ' Expand columns and rows to display the outline at levels 0 - 2.
   Msgbox "Expand columns and rows to display the outline at levels 0 - 2."
   [A].OutlineColumnsToLevel 2
   [A].OutlineRowsToLevel 2

   ' Clear all outline levels for the current sheet.
   Msgbox "Clear the outline."
   [A].ClearOutline
End Sub
```

```
' Example: QuerySortDefineKey, Refresh, RemoveSelectField,
'          SetRecordsLimitMax, SortData, SortReset, and Update
'          methods
Sub TableManners
   ' Open the data table EMPLOYEE.DBF as a new document.
   ' The data is stored in cells A:A1..A:F11.
   MessageBox "Create a new document for data in an external database."
   CurrentApplication.OpenDocument _
      "E:\data\123\Employee.dbf",,"dBase (DBF)",,False,True,True,,

   ' Set up ranges for the query.
   MessageBox "Define source and output ranges for a query."
   ' Assign the range name "source" to cells A:A1..A:F11.
   CurrentDocument.CreateRangeName _
      "source",[A:A1..A:F11]
   ' Assign the range name "outputrange" to cells A:A15..A:F26.
   CurrentDocument.CreateRangeName _
      "outputrange",[A:A15..A:F26]

   ' Create the query named QueryB.
   MessageBox "Create the query and set its working properties."
   CurrentDocument.NewQuery "QueryB"
   ' Set some properties for the new query.
   ' Assign the source data to the workbook range named "source".
   [QueryB].BaseSourceTable = "source"
   [QueryB].ExtractingUniqueRecords = False
   [QueryB].SetRecordsLimitMax True,1000
   ' Assign the query output to the workbook range named "outputrange".
   [QueryB].OutputLocation = "outputrange"
   ' Force a refresh of the output range.
   [QueryB].Refresh

   ' Sort data in the source and output ranges on the
   ' field named "Last".
   MessageBox "Sort the data on the field named Last."
   [QueryB].SortData "Last", $Ascend

   ' Define a sort field for the query on the field named "DEPTNUM".
   MessageBox "Define a new sort key for the data and resort."
   [QueryB].QuerySortDefineKey "DEPTNUM", 1, $Ascend
   ' Force the sort using the new sort key.
   [QueryB].SortData

   ' Reset all sort fields defined for the query.
   MessageBox "Reset all sort keys."
   [QueryB].SortReset

   ' Remove the field named DEPTNUM from the output range.
   MessageBox "Remove the field DEPTNUM from the query and output range."
   [QueryB].RemoveSelectField "DEPTNUM"

   ' Update data in the query.
   ' NOTE -- The query cannot contain computed fields or
   ' aggregate fields NOT in the source table.
   MessageBox "Update the data in the query."
   [QueryB].AllowsUpdates = True
   [QueryB].Update

   ' Remove the field named EMPID from the output range.
```

```
   MessageBox "Remove the field EMPID from the query and output range."
   [QueryB].RemoveSelectField "EMPID"

End Sub
```

```
' Example: Quit method
' Exit the application, without saving changed documents.
CurrentApplication.Quit False
```

```
' Example: RangeCombine method
' This example creates a new workbook document, adds some data to it,
' and saves it to a file called Testdoc1.123.
' The example then creates a second new workbook document, and uses
' RangeCombine to combine the data previously saved in testdoc1.123.
' First create two variables for the documents and get the default path
    Dim TestDocument1 As Document
    Dim TestDocument2 As Document
    Dim DefaultPath As String
'Get the value of the default path
    DefaultPath = CurrentApplication.DefaultPath
' Open a new document and call it TestDocument1.
    Set TestDocument1 = CurrentApplication.NewDocument("TestDoc1")
' Add some data to cells A:A1 through A:A3 in TestDocument1.
    [A:A1].Select
    Selection.Contents = "Test document for RangeCombine script example"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "10"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "20"
' Save the contents of TestDocument1 to the file testdoc1.123
' in the default path.
    TestDocument1.SaveAs DefaultPath & "Testdoc1.123"
' Close TestDocument1
    TestDocument1.Close
' Open a new document and call it TestDocument2.
    Set TestDocument2 = CurrentApplication.NewDocument("TestDoc2")
' Use RangeCombine method to combine the data saved in testdoc1.123.
    [A:A1].RangeCombine DefaultPath & "Testdoc1.123"
' Delete the test file
    Kill DefaultPath & "Testdoc1.123"
```

```
' Example: RangeCombineText method
'This example first opens and writes some data to a text file. It then creates
'a new 1-2-3 workbook and uses the RangeCombineText to insert the data
'that was written to the text file.
' First dimension some variables
    Dim TestDocument As Document
    Dim DefaultPath As String
    Dim fileNumber As Integer
'Get the value of the default path
    DefaultPath = [].DefaultPath
'Create a text file and put some data in it
    fileNumber% = Freefile

    Open DefaultPath & "rct.txt" For Output As fileNumber%
    Write #fileNumber%,"RangeCombineText example input file"
    Write #fileNumber%,10,20,30
    Write #fileNumber%,40,50,60
    Write #fileNumber%,70,80,90
    Close fileNumber%
' Open a new workbook and call it TestDoc
    Set TestDocument = CurrentApplication.NewDocument("TestDoc")
' Use the RangeCombineText method to combine the data saved in rct.txt
    [A:A1].RangeCombineText "rct.txt",DefaultPath,,","
'Use the MessageBox statement to display a
'message asking if you want to delete the test documents and test file.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers 'with the MessageBox statement.
    answer% = Messagebox("Do you want to delete the test documents
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document and delete the test file
        CurrentDocument.Close False
        Kill DefaultPath & "rct.txt"
    End If
```

```
'Example: RangeExtract method
'This example creates a test document and fills the range A:B1..A:B3 with some data.
'It then uses the RangeExtract method to copy the range to a test file.
'First create two variables for the documents and get the default path
    Dim TestDocument1 As Document
    Dim TestDocument2 As Document
    Dim DefaultPath As String
'Get the value of the default path
    DefaultPath = CurrentApplication.DefaultPath
' Open a new document and call it TestDocument1.
    Set TestDocument1 = CurrentApplication.NewDocument("TestDoc1")
' Add some data to cells A:A1 through A:A3 in TestDocument1.
    [A:B1].Select
    Selection.Contents = "Test document for RangeExtract script example"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "10"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "20"
' Use the RangeExtract method to extract the contents of A:A1..A:A3 to the file
"Testdoc1.123"
    [A:B1..A:B3].RangeExtract DefaultPath & "Testdoc2.123"
    'Note that RangeExtract always extracts to the range starting in cell A:A1 of the
new file.
'Open Testdoc2.123 to check that the range got extracted.
    CurrentApplication.OpenDocument DefaultPath & "Testdoc2.123"
```

```
' Example: RangeFill method
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
'Fill A1..A10 with even numbers
    [A1..A10].RangeFill 2,2,100,$Number

'Use the MessageBox statement to display a
'message asking if you want to close the test document.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document
        CurrentDocument.Close False
    End If
```

```
' Example: RangeSortDefineKey, SortResetKeys, and Sort methods
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Add some data to cells A:A1 through A:B6 in TestDocument.
    [A:A1].Contents = "Test document for script example"
    [A:A2].Contents = "25"
    [A:A3].Contents = "4"
    [A:A4].Contents = "93"
    [A:A5].Contents = "41"
    [A:A6].Contents = "56"
    [A:B2].Contents = "325"
    [A:B3].Contents = "83"
    [A:B4].Contents = "12"
    [A:B5].Contents = "71"
    [A:B6].Contents = "256"
'Define A:A2 as the sort key and sort the data in A1..B6
    Messagebox("Define A2 as the sort key and sort the range.")
    TestDocument.RangeSortDefineKey 0, [A:A2], $Ascend
    [A:A2..A:B6].Sort
'Reset the keys, define B2 as the sort key and sort the range
    Messagebox("Reset keys, define B2 as the sort key, and sort the range.")
    TestDocument.SortResetKeys
    TestDocument.RangeSortDefineKey 0, [A:B2], $Ascend
    [A:A2..A:B6].Sort
```

```
' Example: RangeValue method
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
'Fill A1..A10 with even numbers
    [A1..A10].RangeFill 2,2,100,$Number
'Use @sum to sum the values
    [A12].Select
    Selection.Contents = "@sum(A1..A10)"
'Use RangeValue to copy the value in A12 to C12
    [A12].RangeValue [C12]

'Use the MessageBox statement to display a
'message asking if you want to close the test document.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document
        CurrentDocument.Close False
    End If
```

```
' Example: RecalcRange method
' Recalculates the formulas in A1..C10 by rows.
[A1..C10].RecalcRange $Rows
```

```
' Example: QuerySortDefineKey, Refresh, RemoveSelectField,
'          SetRecordsLimitMax, SortData, SortReset, and Update
'          methods
Sub TableManners
   ' Open the data table EMPLOYEE.DBF as a new document.
   ' The data is stored in cells A:A1..A:F11.
   Msgbox "Create a new document for data in an external database."
   CurrentApplication.OpenDocument _
      "E:\data\123\Employee.dbf",,"dBase (DBF)",,False,True,True,,

   ' Set up ranges for the query.
   Msgbox "Define source and output ranges for a query."
   ' Assign the range name "source" to cells A:A1..A:F11.
   CurrentDocument.CreateRangeName _
      "source",[A:A1..A:F11]
   ' Assign the range name "outputrange" to cells A:A15..A:F26.
   CurrentDocument.CreateRangeName _
      "outputrange",[A:A15..A:F26]

   ' Create the query named QueryB.
   Msgbox "Create the query and set its working properties."
   CurrentDocument.NewQuery "QueryB"
   ' Set some properties for the new query.
   ' Assign the source data to the workbook range named "source".
   [QueryB].BaseSourceTable = "source"
   [QueryB].ExtractingUniqueRecords = False
   [QueryB].SetRecordsLimitMax True,1000
   ' Assign the query output to the workbook range named "outputrange".
   [QueryB].OutputLocation = "outputrange"
   ' Force a refresh of the output range.
   [QueryB].Refresh

   ' Sort data in the source and output ranges on the
   ' field named "Last".
   Msgbox "Sort the data on the field named Last."
   [QueryB].SortData "Last", $Ascend

   ' Define a sort field for the query on the field named "DEPTNUM".
   Msgbox "Define a new sort key for the data and resort."
   [QueryB].QuerySortDefineKey "DEPTNUM", 1, $Ascend
   ' Force the sort using the new sort key.
   [QueryB].SortData

   ' Reset all sort fields defined for the query.
   Msgbox "Reset all sort keys."
   [QueryB].SortReset

   ' Remove the field named DEPTNUM from the output range.
   Msgbox "Remove the field DEPTNUM from the query and output range."
   [QueryB].RemoveSelectField "DEPTNUM"

   ' Update data in the query.
   ' NOTE -- The query cannot contain computed fields or
   ' aggregate fields NOT in the source table.
   Msgbox "Update the data in the query."
   [QueryB].AllowsUpdates = True
   [QueryB].Update

   ' Remove the field named EMPID from the output range.
```

```
   Msgbox "Remove the field EMPID from the query and output range."
   [QueryB].RemoveSelectField "EMPID"

End Sub
```

```
' Example: CreateRangeName, OpenDocument, NewQueryTable,
'          RefreshOutput and RefreshQuery methods
' This example sets up an Approach query table, changes its source data,
' and then refreshes the query table and output range data.

Sub RefreshTables
   ' Open the document EMPLOYEE.123.
   ' The data is stored in cells A:A1..A:F11.
   Msgbox "Open the document EMPLOYEE.123."
   CurrentApplication.OpenDocument "E:\data\123\Employee.123",, _
      "1-2-3 Workbooks (123;WK*)",, False, True, True,,

   ' Set up ranges for the query table.
   Msgbox "Define source and output ranges for a query table."
   ' Assign the range name "source" to cells A:A1..A:F11.
   CurrentDocument.CreateRangeName "source", [A:A1..A:F11]
   ' Assign the range name "outputrange" to cells A:A15..A:F26.
   CurrentDocument.CreateRangeName "outputrange", [A:A15..A:F26]

   ' Declare a variable for a query table.
   Dim QT1 As QueryTable
   ' Assign the Approach query table to the source range "source"
   ' and to the output range named "outputrange".
   Msgbox "Create a query table based on workbook ranges."
   Set QT1 = [A].NewQueryTable(252,4176,252,4176, _
      "ApproachWorksheet",,,,,,, [SOURCE],[OUTPUTRANGE])

   ' Change some data in the source range.
   Msgbox "Change data in a cell in the source range."
   [A:C2].Contents = "Bob"
   ' Refresh the data in the query table and (automatically) the output range.
   Msgbox "Refresh the query table and output range."
   QT1.RefreshQuery

   ' NOTE -- If you change data directly in the query table, you can use the
   ' RefreshOutput method to refresh the contents of the output range.
   QT1.RefreshOutput
End Sub
```

```
' Example: RemoveFromSelection method
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Create two ranges and select them.
    Messagebox("Create two ranges and select them.")
    Dim range1 As Range, range2 As Range
    Set range1 = Bind("A:A1..A:A10")
    Set range2 = Bind("A:B10..A:B20")
' Select A:A1..A:A10, then add A:B10..A:B20 to the selection.
    range1.Select
    range2.AddToSelection
' Remove range1 from the selection
    Messagebox("Remove the first range from the selection.")
    range1.RemoveFromSelection

'Use the MessageBox statement to display a
'message asking if you want to close the test document.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document
        CurrentDocument.Close False
    End If
```

```
' Example: QuerySortDefineKey, Refresh, RemoveSelectField,
'          SetRecordsLimitMax, SortData, SortReset, and Update
'          methods
Sub TableManners
   ' Open the data table EMPLOYEE.DBF as a new document.
   ' The data is stored in cells A:A1..A:F11.
   Msgbox "Create a new document for data in an external database."
   CurrentApplication.OpenDocument _
      "E:\data\123\Employee.dbf",,"dBase (DBF)",,False,True,True,,

   ' Set up ranges for the query.
   Msgbox "Define source and output ranges for a query."
   ' Assign the range name "source" to cells A:A1..A:F11.
   CurrentDocument.CreateRangeName _
      "source",[A:A1..A:F11]
   ' Assign the range name "outputrange" to cells A:A15..A:F26.
   CurrentDocument.CreateRangeName _
      "outputrange",[A:A15..A:F26]

   ' Create the query named QueryB.
   Msgbox "Create the query and set its working properties."
   CurrentDocument.NewQuery "QueryB"
   ' Set some properties for the new query.
   ' Assign the source data to the workbook range named "source".
   [QueryB].BaseSourceTable = "source"
   [QueryB].ExtractingUniqueRecords = False
   [QueryB].SetRecordsLimitMax True,1000
   ' Assign the query output to the workbook range named "outputrange".
   [QueryB].OutputLocation = "outputrange"
   ' Force a refresh of the output range.
   [QueryB].Refresh

   ' Sort data in the source and output ranges on the
   ' field named "Last".
   Msgbox "Sort the data on the field named Last."
   [QueryB].SortData "Last", $Ascend

   ' Define a sort field for the query on the field named "DEPTNUM".
   Msgbox "Define a new sort key for the data and resort."
   [QueryB].QuerySortDefineKey "DEPTNUM", 1, $Ascend
   ' Force the sort using the new sort key.
   [QueryB].SortData

   ' Reset all sort fields defined for the query.
   Msgbox "Reset all sort keys."
   [QueryB].SortReset

   ' Remove the field named DEPTNUM from the output range.
   Msgbox "Remove the field DEPTNUM from the query and output range."
   [QueryB].RemoveSelectField "DEPTNUM"

   ' Update data in the query.
   ' NOTE -- The query cannot contain computed fields or
   ' aggregate fields NOT in the source table.
   Msgbox "Update the data in the query."
   [QueryB].AllowsUpdates = True
   [QueryB].Update

   ' Remove the field named EMPID from the output range.
```

```
   Msgbox "Remove the field EMPID from the query and output range."
   [QueryB].RemoveSelectField "EMPID"

End Sub
```

```
' Example: Background, BackColor, Colors properties; AddVersion,
'          CreateRangeName, DeleteVersion, DeleteVersionGroup, NewVersion,
'          NewVersionGroup, RemoveVersion methods
' This example creates 2 range names and a version for each.
' It then puts the versions in a version group.
' First, create the ranges.
CurrentDocument.CreateRangeName "North Sales", [A:B2..A:B10]
CurrentDocument.CreateRangeName "South Sales", [A:C2..A:C10]
' Next, make a second version for each range.
[North Sales].NewVersion("Version 2")
[South Sales].NewVersion("Version 2")
' Do something to make Version 2 different. For example, set the background color.
Set [North Sales].Background.BackColor = CurrentApplication.Colors("light yellow")
Set [South Sales].Background.BackColor = CurrentApplication.Colors("pale blue")
' Create a version group and add the versions to it.
CurrentDocument.NewVersionGroup("Version Group A")
[Version Group A].AddVersion "North Sales", "Version 2"
[Version Group A].AddVersion "South Sales", "Version 2"
' After you're finished with the version, you can remove it or delete it.
' If necessary, remove a version from the version group.
[Version Group A].RemoveVersion "South Sales"
' If necessary, delete the version.
[South Sales.Version 2].DeleteVersion
' When finished, you can also delete Version Group A from the document.
[Version Group A].DeleteVersionGroup
```

```
' Example: RenameNamedStyle, RevertToNamedStyle, RevertToStyle,
'          SetGalleryStyle, and StyleFontReset methods
Sub SetStyles
   ' Create three labels (one for each named style):
   ' Style1, Style2, and Style3.
   Msgbox "Create labels for styles Style1, Style2, and Style3."
   [A:A4].Select
   Selection.Contents = "Style1"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style2"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style3"

   ' Make each style distinctive.
   Msgbox "Make each style distinctive."
   [A:A4].Select
   Selection.Font.FontColor.ColorName = "blue"
   Selection.Font.Bold = True
   Selection.Background.BackColor.ColorName = "ice blue"
   Selection.DefineNamedStyle "Style1"
   Selection.StyleName = "Style1"
   [A:A5].Select
   Selection.Font.Italic = True
   Selection.Font.FontColor.ColorName = "red"
   Selection.Background.BackColor.ColorName = "blush"
   Selection.DefineNamedStyle "Style2"
   Selection.StyleName = "Style2"
   [A:A6].Select
   Selection.Font.DoubleUnderline = True
   Selection.Font.Size = 14
   Selection.Font.FontColor.ColorName = "dark green"
   Selection.Background.BackColor.ColorName = "pale green"
   Selection.DefineNamedStyle "Style3"
   Selection.StyleName = "Style3"

   ' Make cell A:A6 bold, unlike its named style Style3 which is not bold.
   Msgbox "Change one font property locally in cell A:A6 (Style3)."
   [A:A6].Select
   Selection.Font.Bold = True
   ' Now have all style sttributes in cell A:A6 revert to those defined for
   ' the named style Style3. Cell A:A6 loses its bold attrribute.
   Msgbox "Revert all style attributes to those defdined for Style3."
   Selection.RevertToNamedStyle
   ' Rename Style3 to Style3a.
   Msgbox "Rename Style3 to Style3a."
   Selection.RenameNamedStyle "Style3","Style3a"
```

```
    ' Remove all named style attributes from the selection;
    ' restore defaults for the sheet.
    Msgbox "Restore default font attributes."
    [A:A4..A:A6].Stylefontreset

    ' Add a custom border from the gallery.
    Msgbox "Add a custom border from the gallery."
    [A:A4..A:A6].Select
    Selection.SetGalleryStyle $Picture1
End Sub
```

```
' Example: ReportVersion, Version, and Versions methods
Sub VersionDemo1
   ' Set up a range of data to support three versions.
   MessageBox "Set up a range of data to support three versions."
   [A:A3].Contents = "Item"
   [A:B3].Contents = "Amount"
   [A:A3..A:B3].Font.Bold = True
   [A:A4].Contents = "Income"
   [A:A5].Contents = "Expenses"
   [A:A6].Contents = "Result"
   [A:B4].Contents = "45000"
   [A:B5].Contents = "42000"
   [A:B6].Contents = "+B4-+B5"
   [A:B4..A:B6].Format "US Dollar",0
   CurrentDocument.CreateRangeName "BOTTOMLINE",[A:A3..A:B6]
   [BOTTOMLINE].NewVersion "BestCase"
   [BOTTOMLINE.BestCase].Description = "This is the best scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "40000"
   [BOTTOMLINE.Original].MakeCurrent
   [BOTTOMLINE].NewVersion "WorstCase"
   [BOTTOMLINE.WorstCase].Description = "This is the worst case scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "47000"
   ' Use the Version method to access properties for the new versions.
   MessageBox "Use the Version method to access properties for the new versions."
   ' Declare a variable as a Version object.
   Dim returnRangeVersion As Version
   ' Select the range containing the versions.
   [BOTTOMLINE].Goto
   [BOTTOMLINE].Select
   ' Assign the variable to one of the versions in the range.
   Set returnRangeVersion = Selection.Version("BestCase")
   ' Declare a variable to hold the version ID of the selected version.
   Dim rangeVersionVersionID As Long
   ' Get the VersionID of the version.
   rangeVersionVersionID = returnRangeVersion.VersionID
   Print "VersionID for version BestCase = " rangeVersionVersionID
   ' Use the Versions method to access in a Versions collection.
   MessageBox "Use the Versions method to get collection properties."
   ' Declare a variable as a Versions collection object.
   Dim returnRangeVersions As versions
   ' Select the range containing the versions.
   [BOTTOMLINE].Goto
   [BOTTOMLINE].Select
   ' Assign the variable to collection of versions in the selected range.
   Set returnRangeVersions = Selection.Versions
   ' Declare a variable to hold the value of the Count property.
   Dim rangeVersionsCount As Long
   ' Get the number of versions in the selected range.
   rangeVersionsCount = Selection.Versions.Count
   Print "Number of versions in range BOTTOMLINE = " rangeVersionsCount
   ' Generate a report about versions in the selected range.
   MessageBox "Generate a report about versions in the selected range."
```

```
   [BOTTOMLINE].ReportVersion "BestCase;Original;",[A:A20],True,True,$Column
End Sub
```

```
' Example: ReadOnly property; Close, NewDocument, OpenDocument,
'          ReservationGet, ReservationReleased, Save, SaveAs methods
' Open a file and attempt to get a reservation it.
' If successful, modify the file, save it, and release the reservation.
' First, create a test file.
CurrentApplication.NewDocument "My Doc #1"
CurrentDocument.SaveAs "MYFILE1.123"
CurrentDocument.Close
' Open the file and attempt to get its reservation.
CurrentApplication.OpenDocument "MYFILE1.123"
If CurrentDocument.ReadOnly = True Then
   CurrentDocument.ReservationGet
End If
' Check that the reservation was obtained before modifying the file.
If CurrentDocument.ReadOnly = False Then
   ' Reservation obtained.
   ' Write something to the document file ...
   [A:B2].Contents = Today & " version"
   CurrentDocument.Save
   ' When finished with the file, release the reservation.
   CurrentDocument.ReservationReleased
Else
   ' Reservation denied.
   ' Don't write to the document file for now.
End If
```

```
' Example: ResetColumnWidth and ResetRowHeight methods
Sub WidthAndHeight
   ' Set the column width for cell A:B2 to 25 characters.
   Msgbox "Set the column width for cell A:B2 to 25 characters."
   [A:B2].ColumnWidth = 25

   ' Set the row height for cell A:B2 to 25 points.
   Msgbox "Set the row height for cell A:B2 to 25 points."
   [A:B2].RowHeight = 25

   ' Reset the column width for A:B2 to the default column width.
   Msgbox "Reset the column width for A:B2 to the default column width."
   [A:B2].ResetColumnWidth
   ' Reset the row height for A:B2 to the default row height.
   Msgbox "Reset the row height for A:B2 to the default row height."
   [A:B2].ResetRowHeight
End Sub
```

```
' Example: ResetFieldAggregates, AddSelectField, RemoveSelectField methods
Sub Aggregates
   ' Set a database field to be a sum field.
   ' Create an aggregate field column in the query table named Query_1
   ' to add a column of values.

   ' Declare a string variable to hold the names of SelectFields
   ' in the query.
   Dim returnSelectFields As String

   ' Create some data in the sheet for the query.
   [A:B2].Contents = "Employee"
   [A:C2].Contents = "LastName"
   [A:D2].Contents = "AmountSold"
   [A:B3].Contents = "Employee1"
   [A:C3].Contents = "LastName1"
   [A:D3].Contents = "100"
   [A:B4].Contents = "Employee2"
   [A:C4].Contents = "LastName2"
   [A:D4].Contents = "200"
   [A:B5].Contents = "Employee3"
   [A:C5].Contents = "LastName3"
   [A:D5].Contents = "300"

   ' Create the query based on the data in range A:B2..A:D5.
   Msgbox "Create the query based on the data in range A:B2..A:D5."
   CurrentDocument.NewQuery "Query 1", "A:B2..A:D5"

   ' Run the query to generate ouput.
   Msgbox "Run the query to generate ouput."
   [Query 1].OutputLocation = "A:B10"
   [Query 1].Refresh

   ' Report the names of SelectFields in the query as they are
   ' removed and restored.
   returnSelectFields =[Query 1].SelectFields
   Msgbox "Initial SelectFields in the query = " & returnSelectFields

   ' Remove the SelectField named "Employee".
   Msgbox "Remove the SelectField named Employee."
   [Query 1].RemoveSelectField "Employee"
   returnSelectFields =[Query 1].SelectFields
   Msgbox "Current SelectFields in the query = " & returnSelectFields

   ' Restore the SelectField named "Employee".
   Msgbox "Restore the SelectField named Employee."
   [Query 1].AddSelectField "Employee"
   returnSelectFields =[Query 1].SelectFields
   Msgbox "Current SelectFields in the query = " & returnSelectFields

   ' Set a field in the query (AmountSold) to be an aggregate.
   Msgbox "Set a field in the query (AmountSold) to be an aggregate."
   [Query 1].FieldAggregateType "AmountSold", $SUM
   [Query 1].Refresh

   ' Remove the field aggregate formula.
   Msgbox "Remove the field aggregate formula."
   [Query 1].ResetFieldAggregates
   ' Force a refresh.
```

```
   [Query 1].Refresh
End Sub
```

```
' Example: Close and ResetMenuBar methods;
           PreClose and PostClose event handlers


' In the following example, you use the PreClose and PostClose events
' on the document to reset the menu bar whenever the document is closed.

' In the Globals section, declare a Boolean flag
' for blocking the close until the reset is done.
Dim menuBarDefault As Variant

' Bind the following two handlers to the Document object events.
Function PreClose(source As Document, p1 As Variant) As Variant
   If menuBarDefault = True Then
      ' Continue the close. The menu bar has already been reset.
      PreClose = $Continue
   Else
      ' Block the close and raise the PostClose event.
      PreClose = $Block
   End If
End Function

' PostClose is raised only when PreClose returns $Block.
Sub PostClose(Source As Document, P1 As Variant)
   ' Reset the menu bar.
   CurrentApplication.ResetMenuBar
   ' Allow the Close method to proceed in the PreClose handler.
   menuBarDefault = True
   ' Now actually close the document.
   source.Close
End Sub
```

```
' Example: ResetColumnWidth and ResetRowHeight methods
Sub WidthAndHeight
   ' Set the column width for cell A:B2 to 25 characters.
   Msgbox "Set the column width for cell A:B2 to 25 characters."
   [A:B2].ColumnWidth = 25

   ' Set the row height for cell A:B2 to 25 points.
   Msgbox "Set the row height for cell A:B2 to 25 points."
   [A:B2].RowHeight = 25

   ' Reset the column width for A:B2 to the default column width.
   Msgbox "Reset the column width for A:B2 to the default column width."
   [A:B2].ResetColumnWidth
   ' Reset the row height for A:B2 to the default row height.
   Msgbox "Reset the row height for A:B2 to the default row height."
   [A:B2].ResetRowHeight
End Sub
```

```
' Example: Reshape method
' Changes the coordinates of the selected range from "A1..A10' to "B1..C20".
[A1..A10].Select
Selection.Reshape[B1..C20]
```

```
' Example: Resize method
' Create a rectangle and resize it.
[A].NewRectangle 1830,1935,3735,3000
MessageBox "Make the rectangle smaller"
[Rectangle 1].Resize 150,150
```

```
' Example: Resize, Restore, ToBack, and ToFront methods
Sub StackAndSize
   Msgbox "Create three nested rectangles with different background colors."
   ' Create one large, red rectangle in the current sheet.
   .NewRectangle 465,525,2445,1935
   [Rectangle 1].Background.BackColor.ColorName = "red"
   [Rectangle 1].Background.Pattern = $FineCrossHatch

   ' Create a smaller green rectangle at the same origin point.
   .NewRectangle 465,525,2000,1500
   [Rectangle 2].Background.BackColor.ColorName = "neon green"
   [Rectangle 2].Background.Pattern = $FineCrossHatch

   ' Create a smaller turquoise rectangle at the same origin point.
   .NewRectangle 465,525,1200,900
   [Rectangle 3].Background.BackColor.ColorName = "turquoise"
   [Rectangle 3].Background.Pattern = $FineCrossHatch

   ' Move the first rectangle from the bottom of the stack to the top.
   Msgbox "Move the first rectangle from the bottom of the stack to the top."
   [Rectangle 1].ToFront
   ' Move the first rectangle back to the bottom of the stack.
   Msgbox "Move the first rectangle back to the bottom of the stack."
   [Rectangle 1].ToBack

   ' Make the first rectangle bigger.
   Msgbox "Make the first rectangle bigger."
   [Rectangle 1].Resize 3000, 2500

   ' Make the current window smaller.
   Msgbox "Make the current window smaller."
   CurrentWindow.Resize 300, 200

   ' Restore the current window to the default window size.
   Msgbox "Restore the current window to the default window size."
   CurrentWindow.Restore
End Sub
```

```
' Example: RetrieveFileFromInternet method
' Retrieve an HTML file from the Internet and open it.
' Open the file by FTP URL and copy it to the user's TEMP directory.
Dim filename As String
filename = .RetrieveFileFromInternet("ftp://ftp.support.lotus.com/pub/index.html",,,,
_
 "myproxy", myproxyport, 2)
CurrentApplication.OpenDocument filename
```

```
' Example: RetrievePrintSettings method
' Retrieve print settings from a named page settings (.AL3) file
' that was created in an earlier release of 1-2-3.
' Then print the current sheet.
CurrentDocument.RetrievePrintSettings "D:\LOTUS\WORK\123\PRTSET1.AL3"
CurrentDocument.CurrentPrintSettings.PrintWhat = $CurrentSheet
CurrentApplication.Print
```

```
' Example: RenameNamedStyle, RevertToNamedStyle, RevertToStyle,
'          SetGalleryStyle, and StyleFontReset methods
Sub SetStyles
   ' Create three labels (one for each named style):
   ' Style1, Style2, and Style3.
   Msgbox "Create labels for styles Style1, Style2, and Style3."
   [A:A4].Select
   Selection.Contents = "Style1"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style2"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style3"

   ' Make each style distinctive.
   Msgbox "Make each style distinctive."
   [A:A4].Select
   Selection.Font.FontColor.ColorName = "blue"
   Selection.Font.Bold = True
   Selection.Background.BackColor.ColorName = "ice blue"
   Selection.DefineNamedStyle "Style1"
   Selection.StyleName = "Style1"
   [A:A5].Select
   Selection.Font.Italic = True
   Selection.Font.FontColor.ColorName = "red"
   Selection.Background.BackColor.ColorName = "blush"
   Selection.DefineNamedStyle "Style2"
   Selection.StyleName = "Style2"
   [A:A6].Select
   Selection.Font.DoubleUnderline = True
   Selection.Font.Size = 14
   Selection.Font.FontColor.ColorName = "dark green"
   Selection.Background.BackColor.ColorName = "pale green"
   Selection.DefineNamedStyle "Style3"
   Selection.StyleName = "Style3"

   ' Make cell A:A6 bold, unlike its named style Style3 which is not bold.
   Msgbox "Change one font property locally in cell A:A6 (Style3)."
   [A:A6].Select
   Selection.Font.Bold = True
   ' Now have all style sttributes in cell A:A6 revert to those defined for
   ' the named style Style3. Cell A:A6 loses its bold attrribute.
   Msgbox "Revert all style attributes to those defdined for Style3."
   Selection.RevertToNamedStyle
   ' Rename Style3 to Style3a.
   Msgbox "Rename Style3 to Style3a."
   Selection.RenameNamedStyle "Style3","Style3a"
```

```
    ' Remove all named style attributes from the selection;
    ' restore defaults for the sheet.
    Msgbox "Restore default font attributes."
    [A:A4..A:A6].Stylefontreset

    ' Add a custom border from the gallery.
    Msgbox "Add a custom border from the gallery."
    [A:A4..A:A6].Select
    Selection.SetGalleryStyle $Picture1
End Sub
```

```
' Example: RenameNamedStyle, RevertToNamedStyle, RevertToStyle,
'          SetGalleryStyle, and StyleFontReset methods
Sub SetStyles
   ' Create three labels (one for each named style):
   ' Style1, Style2, and Style3.
   Msgbox "Create labels for styles Style1, Style2, and Style3."
   [A:A4].Select
   Selection.Contents = "Style1"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style2"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style3"

   ' Make each style distinctive.
   Msgbox "Make each style distinctive."
   [A:A4].Select
   Selection.Font.FontColor.ColorName = "blue"
   Selection.Font.Bold = True
   Selection.Background.BackColor.ColorName = "ice blue"
   Selection.DefineNamedStyle "Style1"
   Selection.StyleName = "Style1"
   [A:A5].Select
   Selection.Font.Italic = True
   Selection.Font.FontColor.ColorName = "red"
   Selection.Background.BackColor.ColorName = "blush"
   Selection.DefineNamedStyle "Style2"
   Selection.StyleName = "Style2"
   [A:A6].Select
   Selection.Font.DoubleUnderline = True
   Selection.Font.Size = 14
   Selection.Font.FontColor.ColorName = "dark green"
   Selection.Background.BackColor.ColorName = "pale green"
   Selection.DefineNamedStyle "Style3"
   Selection.StyleName = "Style3"

   ' Make cell A:A6 bold, unlike its named style Style3 which is not bold.
   Msgbox "Change one font property locally in cell A:A6 (Style3)."
   [A:A6].Select
   Selection.Font.Bold = True
   ' Now have all style attributes in cell A:A6 revert to those defined for
   ' the named style Style3. Cell A:A6 loses its bold attrribute.
   Msgbox "Revert all style attributes to those defined for Style3."
   Selection.RevertToNamedStyle
   ' Rename Style3 to Style3a.
   Msgbox "Rename Style3 to Style3a."
   Selection.RenameNamedStyle "Style3","Style3a"
```

```
    ' Remove all named style attributes from the selection;
    ' restore defaults for the sheet.
    Msgbox "Restore default font attributes."
    [A:A4..A:A6].Stylefontreset

    ' Add a custom border from the gallery.
    Msgbox "Add a custom border from the gallery."
    [A:A4..A:A6].Select
    Selection.SetGalleryStyle $Picture1
End Sub
```

```
' Example: SameColor method
Sub TestColorObjects
   ' Declare a variable for the first new color object.
   Dim firstColor As Color
   ' Declare a variable for the second new color object.
   Dim secondColor As Color
   ' Declare a variable to hold the return value of the SameColor method.
   Dim colorissame As Variant

   ' Assign the color blue to the first color object.
   Set firstColor = CurrentApplication.Colors("blue")
   ' Assign the color blue to the second color object.
   Set secondColor = CurrentApplication.Colors("blue")
   ' Compare the color of the first color object with that of the second
   ' color object and return True (because the colors are the same).
   colorissame = firstColor.SameColor(secondColor)
   Print colorissame

   ' Here is an efficient way to use SameColor.
   ' If Color1.SameColor(Color2) Then
   '    Print "Color2 is identical to Color1"
   ' End If
End Sub
```

```
' Example: Sheets property; NewDocument, Save and SaveAs methods
' Save a file.
' First create a test file.
Dim document1 As Document
' Open a new document without specifying a file or path.
Set document1 = CurrentApplication.NewDocument
' Add some content to the document.
' For example, put some text into the first sheet name.
document1.Sheets(0).Name = "My sheet 1"
' Save to a file.
document1.SaveAs "d:\lotus\work\123\testopen.123"
' Add more content to the document.
' For example, put some text into the first column head.
[A:A1].Contents = "My Column Head 1"
' Save the changes.
document1.Save
```

```
' Example: ScrollToActiveCell and SetOrigin methods
Sub Origins
   ' Select cell A:A122 and scroll the current window to display that cell.
   Msgbox "Select cell A:A122 and scroll the current window to display it."
   [A].TurnTo
   [A:A122].Select
   .ScrollToActiveCell

   ' Display the specified cell without changing the current selection.
   Msgbox "Display cell (A:F122) without changing the current selection."
   .SetOrigin [A:F122]
End Sub
```

```
' Example: MatchAccent, MatchCase, ReplaceString, and SearchString properties, Replace
and ReplaceAll methods
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Add some data to cells A:A1 through A:A5 in TestDocument.
    [A:A1].Select
    Selection.Contents = "Test document for example"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "grey"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "black and blue"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "Red and Blue"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "blue and white"
' Specify search and replace strings
    CurrentApplication.SearchString = "blue"
    CurrentApplication.ReplaceString = "green"
' Specify search characteristics
    CurrentApplication.MatchAccent = True
    CurrentApplication.MatchCase = False
' Replace the first occurence
    Messagebox("Replace ""blue"" with ""green"" in the first occurence.")
    [A1..A5].Replace
    Messagebox("Replace ""blue"" with ""green"" in all occurences.")
    [A1..A5].ReplaceAll

'Use the MessageBox statement to display a
'message asking if you want to close the test document.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document
        CurrentDocument.Close False
    End If
```

```
'  Example: Select and SelectAll methods
'  Select the range A1..A5.
    Messagebox("Select the range A1..A5.")
    [A1..A5].Select
'  SelectAll
    Messagebox("Use SelectAll to select the active area of the current sheet.")
    CurrentDocument.CurrentSheet.SelectAll
```

```
' Example: Send method and UserLogin method
' Send a document in an e-mail message, using the local mail application.
Dim msg As Document
Set msg = CurrentApplication.NewDocument
[A:A1].Contents = "Message 1"
' Log into the e-mail system.
CurrentApplication.UserLogin "username", "userpassword",
' Send a broadcast message to two recipients.
msg.Send "John X. Smith; Jane X. Doe", "Comment for John; Comment for Jane","Msg
subject", "Msg body",,,, $Broadcast,,,,,,
```

```
' Example: SendMail and UserLogin method
' Send a an e-mail message, using the local mail application.
Dim msg As String
msg = "Body of message"
' Log into the e-mail system.
CurrentApplication.UserLogin "username", "userpassword",
' Send a message to two recipients.
CurrentApplication.SendMail "John X. Smith; Jane X. Doe", "Comment for John; Comment
for Jane", "Msg subject", msg,,,,
```

```
' Example: SetActiveCell method
' Selects a collection of ranges and makes the cell in the top left
' corner of the second range the active cell.
Dim Range1 As Range
Dim Range2 As Range
Set Range1 = Bind("A:A1..A:A10")
Set Range2 = Bind("A:C3..A:H6")
Range1.Select
Range2.AddToSelection
Selection.SetActiveCell
```

```
' Example: SetCellData method
' This example assumes you have an external C routine named "GetAddr" in
' the external library "CellLib.dll".
' Global declaration
Declare Function GetAddr Lib "CellLib.dll" () As Long
' Get external data and enter it into a range.
Dim externpointer As Long
externpointer = GetAddr ()   'Returns a pointer to data to be set.
[A:A1..C:A3].SetCellData externpointer  'Enters values in the range.
```

```
' Example: RenameNamedStyle, RevertToNamedStyle, RevertToStyle,
'          SetGalleryStyle, and StyleFontReset methods
Sub SetStyles
   ' Create three labels (one for each named style):
   ' Style1, Style2, and Style3.
   Msgbox "Create labels for styles Style1, Style2, and Style3."
   [A:A4].Select
   Selection.Contents = "Style1"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style2"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style3"

   ' Make each style distinctive.
   Msgbox "Make each style distinctive."
   [A:A4].Select
   Selection.Font.FontColor.ColorName = "blue"
   Selection.Font.Bold = True
   Selection.Background.BackColor.ColorName = "ice blue"
   Selection.DefineNamedStyle "Style1"
   Selection.StyleName = "Style1"
   [A:A5].Select
   Selection.Font.Italic = True
   Selection.Font.FontColor.ColorName = "red"
   Selection.Background.BackColor.ColorName = "blush"
   Selection.DefineNamedStyle "Style2"
   Selection.StyleName = "Style2"
   [A:A6].Select
   Selection.Font.DoubleUnderline = True
   Selection.Font.Size = 14
   Selection.Font.FontColor.ColorName = "dark green"
   Selection.Background.BackColor.ColorName = "pale green"
   Selection.DefineNamedStyle "Style3"
   Selection.StyleName = "Style3"

   ' Make cell A:A6 bold, unlike its named style Style3 which is not bold.
   Msgbox "Change one font property locally in cell A:A6 (Style3)."
   [A:A6].Select
   Selection.Font.Bold = True
   ' Now have all style sttributes in cell A:A6 revert to those defined for
   ' the named style Style3. Cell A:A6 loses its bold attrribute.
   Msgbox "Revert all style attributes to those defdined for Style3."
   Selection.RevertToNamedStyle
   ' Rename Style3 to Style3a.
   Msgbox "Rename Style3 to Style3a."
   Selection.RenameNamedStyle "Style3","Style3a"
```

```
    ' Remove all named style attributes from the selection;
    ' restore defaults for the sheet.
    Msgbox "Restore default font attributes."
    [A:A4..A:A6].Stylefontreset

    ' Add a custom border from the gallery.
    Msgbox "Add a custom border from the gallery."
    [A:A4..A:A6].Select
    Selection.SetGalleryStyle $Picture1
End Sub
```

```
' Example: SetHorizontalTitle and SetVerticalTitle methods
Sub VertAndHorzTitles
   ' Put labels for the titles in cells A:B1 and A:A2.
   Msgbox "Put labels for the titles in cells A:B1 and A:A2."
   [A].TurnTo
   [A:B1].Contents = "Horizontal title"
   [A:A2].Contents = "VerticalTitle"

   ' Set cell A:B1 as the horizontal title.
   Msgbox "Set cell A:B1 as the horizontal title."
   [A].SetHorizontalTitle [A:B1],1

   ' Set cell A:A2 as the vertical title.
   Msgbox "Set cell A:A2 as the vertical title."
   [A].SetVerticalTitle [A:A2],1
End Sub
```

```
' Example: SetInternetOptions method
' Display the Internet Options dialog box.
CurrentApplication.SetInternetOptions
```

```
' Example: ScrollToActiveCell and SetOrigin methods
Sub Origins
   ' Select cell A:A122 and scroll the current window to display that cell.
   Msgbox "Select cell A:A122 and scroll the current window to display it."
   [A].TurnTo
   [A:A122].Select
   .ScrollToActiveCell

   ' Display the specified cell without changing the current selection.
   Msgbox "Display cell (A:F122) without changing the current selection."
   .SetOrigin [A:F122]
End Sub
```

```
' Example: SetHorizontalTitle and SetVerticalTitle methods
Sub VertAndHorzTitles
   ' Put labels for the titles in cells A:B1 and A:A2.
   Msgbox "Put labels for the titles in cells A:B1 and A:A2."
   [A].TurnTo
   [A:B1].Contents = "Horizontal title"
   [A:A2].Contents = "VerticalTitle"

   ' Set cell A:B1 as the horizontal title.
   Msgbox "Set cell A:B1 as the horizontal title."
   [A].SetHorizontalTitle [A:B1],1

   ' Set cell A:A2 as the vertical title.
   Msgbox "Set cell A:A2 as the vertical title."
   [A].SetVerticalTitle [A:A2],1
End Sub
```

```
' Example: Show method
Sub ShowDocument
   ' Declare a variable for a document object.
   Dim document1 As Document

   ' Open an existing document, setting its MakeVisible parameter to FALSE.
   Msgbox "Open an existing document, setting its MakeVisible parameter to FALSE."
   Set document1 = CurrentApplication.OpenDocument ("Employee.123", _
      "e:\data\123", "1-2-3 (123)",, False, False, True,,)
   Msgbox "The opened document is not visible."
   ' Now show the document.
   Msgbox "Now show the document."
   document1.show
End Sub
```

```
' Example: PageBack, PageForward, ShowAllSheets,
'          ShowSheet, and TurnTo methods
Sub NavigateDoc
   ' Create 10 new sheets after the current sheet in the workbook.
   .NewSheet $After, 10, True
   .PageBack 2
   ' Move forward three sheets in the workbook.
   Msgbox "Move forward three sheets."
   .PageForward 3
   ' Move back one sheet in the workbook.
   Msgbox "Move back one sheet."
   .PageBack

   ' Turn to sheet A in the workbook.
   Msgbox "Turn to sheet A in the workbook."
   [A].TurnTo
   ' Hide sheets B, C, and D.
   Msgbox "Hide sheets B, C, and D."
   [B].HideSheet
   [C].HideSheet
   [D].HideSheet

   ' Unhide sheet B.
   Msgbox "Unhide sheet B."
   [B].ShowSheet

   ' Unhide all hidden sheets (including sheets C and D).
   Msgbox "Unhide all hidden sheets (including sheets C and D)."
   .ShowAllSheets
End Sub
```

```
' Example: PageBack, PageForward, ShowAllSheets,
'          ShowSheet, and TurnTo methods
Sub NavigateDoc
   ' Create 10 new sheets after the current sheet in the workbook.
   .NewSheet $After, 10, True
   .PageBack 2
   ' Move forward three sheets in the workbook.
   Msgbox "Move forward three sheets."
   .PageForward 3
   ' Move back one sheet in the workbook.
   Msgbox "Move back one sheet."
   .PageBack

   ' Turn to sheet A in the workbook.
   Msgbox "Turn to sheet A in the workbook."
   [A].TurnTo
   ' Hide sheets B, C, and D.
   Msgbox "Hide sheets B, C, and D."
   [B].HideSheet
   [C].HideSheet
   [D].HideSheet

   ' Unhide sheet B.
   Msgbox "Unhide sheet B."
   [B].ShowSheet

   ' Unhide all hidden sheets (including sheets C and D).
   Msgbox "Unhide all hidden sheets (including sheets C and D)."
   .ShowAllSheets
End Sub
```

```
' Example: SmartSum method
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Add some data to cells A:A1 through A:A5 in TestDocument.
    [A:A1].Select
    Selection.Contents = "Test document for example"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "10"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "14"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "83"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "27"
'  Use SmartSum method to sum the values in A2..A5.
    [A2..A7].SmartSum
```

```
' Example: QuerySortDefineKey, Refresh, RemoveSelectField,
'          SetRecordsLimitMax, SortData, SortReset, and Update
'          methods
Sub TableManners
   ' Open the data table EMPLOYEE.DBF as a new document.
   ' The data is stored in cells A:A1..A:F11.
   Msgbox "Create a new document for data in an external database."
   CurrentApplication.OpenDocument _
      "E:\data\123\Employee.dbf",,"dBase (DBF)",,False,True,True,,

   ' Set up ranges for the query.
   Msgbox "Define source and output ranges for a query."
   ' Assign the range name "source" to cells A:A1..A:F11.
   CurrentDocument.CreateRangeName _
      "source",[A:A1..A:F11]
   ' Assign the range name "outputrange" to cells A:A15..A:F26.
   CurrentDocument.CreateRangeName _
      "outputrange",[A:A15..A:F26]

   ' Create the query named QueryB.
   Msgbox "Create the query and set its working properties."
   CurrentDocument.NewQuery "QueryB"
   ' Set some properties for the new query.
   ' Assign the source data to the workbook range named "source".
   [QueryB].BaseSourceTable = "source"
   [QueryB].ExtractingUniqueRecords = False
   [QueryB].SetRecordsLimitMax True,1000
   ' Assign the query output to the workbook range named "outputrange".
   [QueryB].OutputLocation = "outputrange"
   ' Force a refresh of the output range.
   [QueryB].Refresh

   ' Sort data in the source and output ranges on the
   ' field named "Last".
   Msgbox "Sort the data on the field named Last."
   [QueryB].SortData "Last", $Ascend

   ' Define a sort field for the query on the field named "DEPTNUM".
   Msgbox "Define a new sort key for the data and resort."
   [QueryB].QuerySortDefineKey "DEPTNUM", 1, $Ascend
   ' Force the sort using the new sort key.
   [QueryB].SortData

   ' Reset all sort fields defined for the query.
   Msgbox "Reset all sort keys."
   [QueryB].SortReset

   ' Remove the field named DEPTNUM from the output range.
   Msgbox "Remove the field DEPTNUM from the query and output range."
   [QueryB].RemoveSelectField "DEPTNUM"

   ' Update data in the query.
   ' NOTE -- The query cannot contain computed fields or
   ' aggregate fields NOT in the source table.
   Msgbox "Update the data in the query."
   [QueryB].AllowsUpdates = True
   [QueryB].Update

   ' Remove the field named EMPID from the output range.
```

```
   Msgbox "Remove the field EMPID from the query and output range."
   [QueryB].RemoveSelectField "EMPID"

End Sub
```

```
' Example: Documents and Changed properties;
' StartPoll, EndPoll, and Save methods; Poll[n] event handler
' Automatically save open documents at regular intervals, using timed poll events.

' Turn on automatic save of open documents. For example,
' you could attach the following autosave enable sub to the Actions menu.
Sub enableAutoSave
   ' Start an indefinite series of poll #2 events occurring once an hour
   ' (every 3,600,000 milliseconds).
   CurrentDocument.StartPoll 2, 3600000, 0
End Sub

' Bind this handler to the Poll2 event.
Sub Poll2(source As Document)
   ' Save all open documents that have changed since their last save.
   Forall doc in CurrentApplication.Documents
      If doc.Changed Then Call doc.Save()
   End Forall
End Sub

' Turn off automatic save of open documents. For example,
' you could attach the following autosave disable sub to the Actions menu.
Sub disableAutoSave
   CurrentDocument.EndPoll 2
End Sub
```

```
' Example: RenameNamedStyle, RevertToNamedStyle, RevertToStyle,
'          SetGalleryStyle, and StyleFontReset methods
Sub SetStyles
   ' Create three labels (one for each named style):
   ' Style1, Style2, and Style3.
   Msgbox "Create labels for styles Style1, Style2, and Style3."
   [A:A4].Select
   Selection.Contents = "Style1"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style2"
   [A].MoveCellPointer $Down,1
   Selection.Contents = "Style3"

   ' Make each style distinctive.
   Msgbox "Make each style distinctive."
   [A:A4].Select
   Selection.Font.FontColor.ColorName = "blue"
   Selection.Font.Bold = True
   Selection.Background.BackColor.ColorName = "ice blue"
   Selection.DefineNamedStyle "Style1"
   Selection.StyleName = "Style1"
   [A:A5].Select
   Selection.Font.Italic = True
   Selection.Font.FontColor.ColorName = "red"
   Selection.Background.BackColor.ColorName = "blush"
   Selection.DefineNamedStyle "Style2"
   Selection.StyleName = "Style2"
   [A:A6].Select
   Selection.Font.DoubleUnderline = True
   Selection.Font.Size = 14
   Selection.Font.FontColor.ColorName = "dark green"
   Selection.Background.BackColor.ColorName = "pale green"
   Selection.DefineNamedStyle "Style3"
   Selection.StyleName = "Style3"

   ' Make cell A:A6 bold, unlike its named style Style3 which is not bold.
   Msgbox "Change one font property locally in cell A:A6 (Style3)."
   [A:A6].Select
   Selection.Font.Bold = True
   ' Now have all style sttributes in cell A:A6 revert to those defined for
   ' the named style Style3. Cell A:A6 loses its bold attrribute.
   Msgbox "Revert all style attributes to those defdined for Style3."
   Selection.RevertToNamedStyle
   ' Rename Style3 to Style3a.
   Msgbox "Rename Style3 to Style3a."
   Selection.RenameNamedStyle "Style3","Style3a"
```

```
    ' Remove all named style attributes from the selection;
    ' restore defaults for the sheet.
    Msgbox "Restore default font attributes."
    [A:A4..A:A6].Stylefontreset

    ' Add a custom border from the gallery.
    Msgbox "Add a custom border from the gallery."
    [A:A4..A:A6].Select
    Selection.SetGalleryStyle $Picture1
End Sub
```

1-2-3: TileHorizontal method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_TILEHORIZONTAL_METHOD_EXSCRIPT ',1)} See example

Tiles all Workbook windows top to bottom.

Syntax

applicationwindow.**TileHorizontal**

Parameters

None

Return values

None

{button ,AL(`H_123_TILE_METHOD_MEMDEF;H_123_TILEVERTICAL_METHOD_MEMDEF',0)} See related topics

```
' Example: SyncSplits, Tile, TileHorizontal, and
'          TileVertical methods
Sub WindowTiles
   ' Create a new window for the curent document.
   Msgbox "Create a new window for the current document."
   CurrentDocument.NewDocWindow

   ' Tile the windows left-right (the default).
   Msgbox "Tile the windows left-right (the default)."
   [ApplicationWindow].Tile

   ' Maximize the current window.
   Msgbox "Maximize the current window"
   .Maximize

   ' Tile the windows left-right explicitly.
   Msgbox "Tile the windows left-right explicitly."
   [ApplicationWindow].TileVertical

   ' Maximize the current window.
   Msgbox "Maximize the current window."
   .Maximize

   ' Tile the windows top-bottom explicitly.
   Msgbox "Tile the windows top-bottom explicitly."
   [ApplicationWindow].TileHorizontal

   ' Synchronize the window splits.
   Msgbox "Synchronize the window splits."
   ' CurrentDocument.Syncsplits

   ' Close the current window.
   Msgbox "Close the current window."
   CurrentWindow.Close

   ' Maximize the current window.
   Msgbox "Maximize the current window."
   .Maximize
End Sub
```

1-2-3: TileVertical method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_TILEVERTICAL_METHOD_EXSCRIPT ',1)} See example

Tiles all Workbook windows side by side.

Syntax

applicationwindow.**TileVertical**

Parameters

None

Return values

None

{button ,AL(`H_123_CASCADE_METHOD_MEMDEF;H_123_TILE_METHOD_MEMDEF;H_123_TILEHORIZONTAL_METHOD_MEMDEF',0)} See related topics

```
' Example: SyncSplits, Tile, TileHorizontal, and
'          TileVertical methods
Sub WindowTiles
   ' Create a new window for the curent document.
   Msgbox "Create a new window for the current document."
   CurrentDocument.NewDocWindow

   ' Tile the windows left-right (the default).
   Msgbox "Tile the windows left-right (the default)."
   [ApplicationWindow].Tile

   ' Maximize the current window.
   Msgbox "Maximize the current window"
   .Maximize

   ' Tile the windows left-right explicitly.
   Msgbox "Tile the windows left-right explicitly."
   [ApplicationWindow].TileVertical

   ' Maximize the current window.
   Msgbox "Maximize the current window."
   .Maximize

   ' Tile the windows top-bottom explicitly.
   Msgbox "Tile the windows top-bottom explicitly."
   [ApplicationWindow].TileHorizontal

   ' Synchronize the window splits.
   Msgbox "Synchronize the window splits."
   ' CurrentDocument.Syncsplits

   ' Close the current window.
   Msgbox "Close the current window."
   CurrentWindow.Close

   ' Maximize the current window.
   Msgbox "Maximize the current window."
   .Maximize
End Sub
```

1-2-3: Tile method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_TILE_METHOD_EXSCRIPT ',1)} See example

Tiles all Workbook windows side by side.

Syntax

applicationwindow.**Tile**

Parameters

None

Return values

None

{button ,AL(`H_123_CASCADE_METHOD_MEMDEF;H_123_TILEHORIZONTAL_METHOD_MEMDEF;H_123_TILEVERTICAL_METHOD_MEMDEF',0)} See related topics

```
' Example: Tile, TileHorizontal, and TileVertical methods
Sub WindowTiles
   ' Create a new window for the curent document.
   MessageBox "Create a new window for the current document."
   CurrentDocument.NewDocWindow
   ' Tile the windows left-right (the default).
   MessageBox "Tile the windows left-right (the default)."
   [ApplicationWindow].Tile
   ' Maximize the current window.
   MessageBox "Maximize the current window"
   .Maximize
   ' Tile the windows left-right explicitly.
   MessageBox "Tile the windows left-right explicitly."
   [ApplicationWindow].TileVertical
   ' Maximize the current window.
   MessageBox "Maximize the current window."
   .Maximize
   ' Tile the windows top-bottom explicitly.
   MessageBox "Tile the windows top-bottom explicitly."
   [ApplicationWindow].TileHorizontal
   ' Close the current window.
   MessageBox "Close the current window."
   CurrentWindow.Close
   ' Maximize the current window.
   MessageBox "Maximize the current window."
   .Maximize
End Sub
```

1-2-3: TimeDifference method

{button ,AL(`H_123_TIMEDIFFERENCE_METHOD_MEMDEF_RT;H_123_DATETIME_CLASS',0)} See list of classes

Returns the difference in seconds between the time of the current object and the time of the specified object.

Syntax

timedifference = **TimeDifference** *time*

Parameters

time

DateTime. The base time to compare with the time of the current object.

Return value

timedifference

Long. The time difference in seconds.

1-2-3: ToBack method

{button ,AL(`H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_APPROACHCONNECTION_CLASS',0)} See list of classes

{button ,AL(`H_123_TOBACK_METHOD_EXSCRIPT ',1)} See example

Sends the specified graphic object to the back of all other overlapping graphic objects. The vertical or horizontal position of the object does not change.

Syntax

drawobject.**ToBack**

Parameters

None

Return values

None

{button ,AL(`H_123_TOFRONT_METHOD_MEMDEF',0)} See related topics

```
'Example: ToBack method
'Selects Rectangle 2 on the current sheet and puts it behind all other graphic objects
on the sheet.
[A].NewRectangle 1770,1035,3765,2715
[Rectangle 1].Select
Selection.Background.BackColor.ColorName = "25% gray"
Selection.Background.Pattern = $SolidBackground
[A].NewRectangle 3225,1470,4530,3045
[Rectangle 2].Select
Selection.Background.BackColor.ColorName = "50% gray"
Selection.Background.Pattern = $SolidBackground
Selection.ToBack
```

1-2-3: ToFront method

{button ,AL(`H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_APPROACHCONNECTION_CLASS',0)} See list of classes

{button ,AL(`H_123_TOFRONT_METHOD_EXSCRIPT ',1)} See example

Sends the specified graphic object to the front of all other overlapping graphic objects. The vertical or horizontal position of the object does not change.

Syntax

drawobject.**ToFront**

Parameters

None

Return values

None

{button ,AL(`H_123_TOBACK_METHOD_MEMDEF',0)} See related topics

```
'Example: ToFront method
'Selects Button 5 on the current sheet and puts it in front of all other graphic
objects on the sheet.
[A].NewButton 540,915,1905,1410
[Button 1].Select
[A].NewButton 2355,945,3495,1425
[Button 2].Select
[A].NewButton 4035,915,5355,1380
[Button 3].Select
[A].NewButton 5880,930,7155,1410
[Button 4].Select
[A].NewButton 930,1860,2070,2250
[Button 5].Select
[A].NewRectangle 315,720,7455,2745
[Rectangle 1].Select
[Button 5].Select
[Button 5].ToFront
```

1-2-3: Transpose method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_TRANSPOSE_METHOD_EXSCRIPT',1)} See example

Copies data in a range to a destination range, transposing rows and columns within the same sheet or across sheets and replacing any copied formulas with their current values.

Syntax

range.**Transpose**(*destinationrange,* [*transposeoption*])

Parameters

*destination*range

Variant. The range to which you are copying. Specify either the entire range or only the first cell.

transposeoption

(Optional) Variant (TransposeEnum enumeration). Specifies how to transpose the data. The following table lists the allowed values for this parameter.

Value	Description
$RowsToColumns	Transposes rows of data in *range* to columns of data in *destinationrange*; default if you omit this parameter.
$ColumnsToSheets	Copies the first column in every sheet of *range* to the first sheet in *destinationrange*; the second column in every sheet of *range* to the second sheet in *destinationrange*; and so on. This argument works only for 3D ranges.
$SheetsToRows	Copies the first row in every sheet of *range* to the first sheet in *destinationrange*; the second row in every sheet of *range* to the second sheet in *destinationrange*; and so on. This argument works only for 3D ranges.

Return values

None

```
' Example: Transpose method
' Open a new document and call it TestDocument.
    Dim TestDocument As Document
    Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Add some data to cells A:A1 through A:B4 in TestDocument.
    [A:A1].Select
    Selection.Contents = "Test document for script example"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "25"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "54"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "93"
    [A:B2].Select
    Selection.Contents = "341"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "956"
    [A].MoveCellPointer $Down,1
    Selection.Contents = "272"
'Transpose
    Messagebox("Transpose the range.")
    [A:A2..A:B4].Transpose [A:A15]

'Use the MessageBox statement to display a
'message asking if you want to delete the test documents and test file.
    Dim boxType As Long, answer As Integer
    BoxType& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document
        CurrentDocument.Close False
    End If
```

1-2-3: TurnTo method

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_TURNTO_METHOD_EXSCRIPT ',1)} See example

Displays the specified sheet.

Syntax

sheet.**TurnTo**

Parameters

None

Return values

None

{button ,AL(`H_123_PAGEBACK_METHOD_MEMDEF;H_123_PAGEFORWARD_METHOD_MEMDEF',0)} See related topics

```
' Example: TurnTo method
' Opens the file BUDGET.123 and displays the sheet named Marketing.
CurrentApplication.OpenDocument "D:\lotus\work\123\budget.123",,False,True,True,,
[Marketing].TurnTo
```

1-2-3: UncheckItem method

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS',0)} See list of classes

{button ,AL(`H_123_CHECKITEM_METHOD_EXSCRIPT',1)} See example

Removes a check mark before the name of a specified item on the menu. This method only works on menu items created using LotusScript.

Syntax

object.**UncheckItem** *position*

Parameters

position

Long. The menu position of the item to check.

Value	Description
Positive integer	The item's position in the menu, counting forward from the beginning. The value 1 means the first position.
Negative integer	The item's position in the menu, counting backward from the end. The value –1 means the last position.

Return values

None

{button ,AL(`H_123_ADDITEM_METHOD_MEMDEF;H_123_DISABLEITEM_METHOD_MEMDEF;H_123_ENABLEITEM_METHOD_MEMDEF;H_123_REMOVEITEM_METHOD_MEMDEF;H_123_REPLACEITEM_METHOD_MEMDEF;H_123_CHECKITEM_METHOD_MEMDEF;H_123_MENUPROMPT_PROPERTY_MEMDEF;H_123_MENUTEXT_PROPERTY_MEMDEF',0)} See related topics

1-2-3: UnGroupSheets method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_UNGROUPSHEETS_METHOD_EXSCRIPT',1)} See example

Clears sheet groups. After you ungroup sheets, the "Grp" indicator no longer appears in the status bar.

Syntax

document.**UnGroupSheets**

Parameters

None

Return values

None

```
' Example: UnGroupSheets method
' Group sheets 0 through 9 (A through J) together,
' and apply the styles and settings in sheet A to the entire group.
CurrentDocument.GroupSheets 0, 9, 0
' Ungroup the sheets
CurrentDocument.UnGroupSheets
```

1-2-3: UnGroup method

{button ,AL(`;H_123_GROUP_CLASS',0)} See list of classes

Ungroups graphic objects so you can manipulate them individually.

Syntax

*object.***UnGroup**

Parameters

None

Return values

None

{button ,AL(`H_123_UNGROUPSHEETS_METHOD_MEMDEF',0)} See related topics

```
' Example: UnhideColumns and UnhideRows methods
Sub HideAndUnhide
   Msgbox "Hide columns B and D."
   ' Hide column B in the current sheet.
   [A:B1..A:B8192].HideColumns
   ' Hide column D in the current sheet.
   [A:D1..A:D8192].HideColumns
   ' Note -- this is equivalent to setting the column property
   ' IsColumnHidden = True

   ' Unhide columns B and D.
   Msgbox "Unhide columns B and D."
   [A:B1..A:B8192].UnHideColumns
   [A:D1..A:D8192].UnHideColumns

   ' Hide rows 2 and 4.
   Msgbox "Hide rows 2 and 4."
   [A:A2..A:IV2].HideRows
   [A:A4..A:IV4].HideRows
   ' Note -- this is equivalent to setting the row property
   ' IsRowHidden = True

   ' Unhide rows 2 and 4.
   Msgbox "Unhide rows 2 and 4."
   [A:A2..A:IV2].UnHideRows
   [A:A4..A:IV4].UnHideRows
End Sub
```

1-2-3: UnhideColumns method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_UNHIDECOLUMNS_EXSCRIPT;',1)} See example

Redisplays all hidden columns in a range. Column widths do not change.

Syntax

range.**UnhideColumns**

Parameters

None

Return values

None

{button ,AL(`H_123_HIDECOLUMNS_METHOD_MEMDEF;H_123_HIDEROWS_METHOD_MEMDEF;H_123_UNHIDEROWS_METHOD_MEMDEF',0)} See related topics

1-2-3: UnhideRows method

{button ,AL(`H_123_Range_Class;',0)} See list of classes

{button ,AL(`H_123_UNHIDEROWS_METHOD_EXSCRIPT',1)} See example

Redisplays all hidden rows in a range. Row heights do not change.

Syntax

range.**UnhideRows**

Parameters

None

Return values

None

{button ,AL(`;H_123_HIDECOLUMNS_METHOD_MEMDEF;H_123_HIDEROWS_METHOD_MEMDEF;H_123_UNHIDECOLUMNS_METHOD_MEMDEF',0)} See related topics

```
' Example: UnhideColumns and UnhideRows methods
Sub HideAndUnhide
   Msgbox "Hide columns B and D."
   ' Hide column B in the current sheet.
   [A:B1..A:B8192].HideColumns
   ' Hide column D in the current sheet.
   [A:D1..A:D8192].HideColumns
   ' Note -- this is equivalent to setting the column property
   ' IsColumnHidden = True

   ' Unhide columns B and D.
   Msgbox "Unhide columns B and D."
   [A:B1..A:B8192].UnHideColumns
   [A:D1..A:D8192].UnHideColumns

   ' Hide rows 2 and 4.
   Msgbox "Hide rows 2 and 4."
   [A:A2..A:IV2].HideRows
   [A:A4..A:IV4].HideRows
   ' Note -- this is equivalent to setting the row property
   ' IsRowHidden = True

   ' Unhide rows 2 and 4.
   Msgbox "Unhide rows 2 and 4."
   [A:A2..A:IV2].UnHideRows
   [A:A4..A:IV4].UnHideRows
End Sub
```

1-2-3: UnloadAddin method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_LOADADDIN_METHOD_EXSCRIPT',1)} See example

Unloads the specified add-in.

Syntax

application.**UnloadAddin** *addinname*

Parameters

addinname

String. The name of the add-in to be unloaded.

Return values

None

Usage

The Application.Addins property is a collection of names of the registered add-ins that you can load or unload. Use the IsAddinLoaded method to determine whether an add-in is loaded.

{button ,AL(`H_123_LOADADDIN_METHOD_MEMDEF;H_123_ISADDINLOADED_METHOD_MEMDEF;H_123_ADDINS_PROPERTY_MEMDEF',0)} See related topics

1-2-3: UpdateDefaultPrintSettings method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Copies the document's CurrentPrintSettings property to the application's DefaultPrintSettings property. This method has the effect of updating the print settings in the Windows registry.

Syntax

document.**UpdateDefaultPrintSettings**

Parameters

None

Return values

None

Usage

The Application.DefaultPrintSettings print style is used for all new workbooks.

{button ,AL(`H_123_USEDEFAULTPRINTSETTINGS_METHOD_MEMDEF;H_123_REDEFINENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_RETRIEVENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NEWNAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NAMEDPRINTSETTINGS_PROPERTY_MEMDEF;H_123_CURRENTPRINTSETTINGS_PROPERTY_MEMDEF',0)} See related topics

1-2-3: Update method

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS;H_123_OLEOBJECT_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS',0)} See list of classes

{button ,AL(`H_123_SORTDATA_METHOD_EXSCRIPT',1)} See example

Updates the object. For a window object, refreshes the window. For a DataQuery object, updates the source table of the query with the modifications made to records in the query table results. For a linked OLE object (including an OLE object inside an OLE object), updates its presentation in the worksheet.

Syntax

object.**Update**

Parameters

None

Return values

None

Usage

For DataQuery objects, this method only works when the DataQuery.AllowsUpdates property is set to True.

For OLE objects, this method only works if the OLEObject.AutoUpdate property is set to $Manual, or the Application.UpdateLinksOnOpenDoc property is set to False.

{button ,AL(`H_REFRESH_METHOD_MEMDEF;H_REFRESHOUTPUT_METHOD_MEMDEF;H_REFRESHQUERY_METHOD_MEMDEF;H_123_AUTOUPDATE_PROPERTY_MEMDEF;H_123_UPDATELINKSONOPENDOC_PROPERTY_MEMDEF;H_123_AUTOREFRESH_PROPERTY_MEMDEF;H_123_ALLOWSUPDATES_PROPERTY_MEMDEF',0)} See related topics

1-2-3: UseDefaultPrintSettings method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_DELETENAMEDPRINTSETTINGS_METHOD_EXSCRIPT',1)} See example

Copies the application's default print settings to the document's CurrentPrintSettings property.

Syntax

document.**UseDefaultPrintSettings**

Parameters

None

Return values

None

{button ,AL(`H_123_UPDATEDEFAULTPRINTSETTINGS_METHOD_MEMDEF;H_123_REDEFINENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_RETRIEVENAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NEWNAMEDPRINTSETTINGS_METHOD_MEMDEF;H_123_NAMEDPRINTSETTINGS_PROPERTY_MEMDEF;H_123_CURRENTPRINTSETTINGS_PROPERTY_MEMDEF',0)} See related topics

1-2-3: UserLogin method

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_SEND_METHOD_EXSCRIPT',1)} See example

Logs the user into the local e-mail application. Call this method before calling the Send or SendMail methods, to prevent the display of the user mail login dialog box.

Syntax

application.**UserLogin** [*username*], [*password*], [*messagecontainerpath*]

Parameters

username

(Optional) String. The user name for the e-mail application.

password

(Optional) String. The password for the e-mail application.

messagecontainerpath

(Optional) String. The full path, including the drive letter, to the user's cc:Mail post office database, if applicable. This parameter is needed only if cc:Mail is the e-mail application.

Return values

None

{button ,AL(`H_123_SEND_METHOD_MEMDEF;H_123_SENDMAIL_METHOD_MEMDEF',0)} See related topics

1-2-3: Verb method

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

Invokes the specified action on an object. The *verb* actions are defined in the registry by the server that manages the object.

Syntax

object.**Verb** *verb*

Parameters

verb

(Optional) String, Integer, or Variant (OLEVerb enumeration). One of the registered verbs supported by the object, such as "Edit" and "Open". Strings are case insensitive, and you can pass the integer equivalent instead. The default is the object's primary verb, which is the action that results when the user double-clicks the object, or presses ENTER when the object is selected.

The following table lists the allowed enumerated values for this parameter. Some objects will only support a subset of these values. If the object doesn't support the specified verb, its server may generate an error.

Enumeration Value	Description
$OLEVerbPrimary	The object's primary verb, normally the action that results when the user double-clicks the object. (Default.)
$OLEVerbShow	Show the object for editing or viewing.
$OLEVerbOpen	Open the object for editing in a separate window.
$OLEVerbHide	Remove the object's user interface (UI) from view.
$OLEVerbUIActivate	Activate the object for in-place editing and show its UI.
$OLEVerbInPlaceActivate	Activate the object for in-place editing, without showing its UI.
$OLEVerbDiscardUndoState	Discard the object's undo state.

Return values

None

1-2-3: VersionGroups method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_VERSIONGROUPS_METHOD_EXSCRIPT ',1)} See example

Returns a collection of version groups matching the specified criteria.

Syntax

versiongroups = *document*.**VersionGroups**([*versiongroupname]*, *[lastmodifier*])

Parameters

versiongroupname

(Optional) String. The name of a version group in the specified file to match. By default, all version groups are included.

lastmodifier

(Optional) String. The name of a user who last modified one or more version groups. If a version group was modified most recently by the name in *lastmodifier,* the VersionGroups method adds that version group to the collection.

Return values

Returns a VersionGroups object.

{button ,AL(`H_123_NEWVERSIONGROUP_METHOD_MEMDEF;H_123_VERSIONGROUP_METHOD_MEMDEF', 0)} See related topics

```
' Example: VersionGroup and VersionGroups methods
Sub VersionDemo2
   ' Set up two ranges of data, each supporting three versions.
   Messagebox "Set up two ranges of data, each supporting three versions."
   Messagebox "The first range is named BOTTOMLINE."
   [A:A3].Contents = "Item"
   [A:B3].Contents = "Amount"
   [A:A3..A:B3].Font.Bold = True
   [A:A4].Contents = "Income"
   [A:A5].Contents = "Expenses"
   [A:A6].Contents = "Result"
   [A:B4].Contents = "45000"
   [A:B5].Contents = "42000"
   [A:B6].Contents = "+B4-+B5"
   [A:B4..A:B6].Format "US Dollar",0
   ' Create the range name BOTTOMLINE and three versions.
   CurrentDocument.CreateRangeName "BOTTOMLINE",[A:A3..A:B6]
   [BOTTOMLINE].NewVersion "BestCase"
   [BOTTOMLINE.BestCase].Description = "This is the best scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "40000"          ' New data for the version.
   [BOTTOMLINE.Original].MakeCurrent
   [BOTTOMLINE].NewVersion "WorstCase"
   [BOTTOMLINE.WorstCase].Description = "This is the worst case scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "47000"          ' New data for the version.
   ' Set up the second range of data and create the versions.
   Messagebox "The second range is named TAXABLES."
   [A:A9].Contents = "Item"
   [A:B9].Contents = "Amount"
   [A:A9..A:B9].Font.Bold = True
   [A:A10].Contents = "Income"
   [A:A11].Contents = "Retirement Account"
   [A:A12].Contents = "Taxable"
   [A:B10].Contents = "45000"
   [A:B11].Contents = "1500"
   [A:B12].Contents = "+B10-+B11"
   [A:B10..A:B12].Format "US Dollar",0
   ' Create the range name TAXABLES and three versions.
   CurrentDocument.CreateRangeName "TAXABLES",[A:A9..A:B12]
   [TAXABLES].NewVersion "BestCase"
   [TAXABLES.BestCase].Description = "This is the best scenario."
   [TAXABLES].VersionBorderVisible = True
   [A:B11].Contents = "2500"          ' New data for the version.
   [TAXABLES.Original].MakeCurrent
   [TAXABLES].NewVersion "WorstCase"
   [TAXABLES.WorstCase].Description = "This is the worst case scenario."
   [TAXABLES].VersionBorderVisible = True
   [A:B11].Contents = "500"           ' New data for the version.
   ' Create a version group named "VersionGroup 1".
   Messagebox "Create a version group named VersionGroup 1."
   CurrentDocument.NewVersionGroup "VersionGroup 1"
   [VersionGroup 1].Description = "Here is a version group."
   ' Add the version BOTTOMLINE.BestCase to the group.
   ' Add the version TAXABLES.BestCase to the group.
   Messagebox "Add versions from BOTTOMLINE and TAXABLES to the group."
   [VersionGroup 1].AddVersion "BOTTOMLINE","BestCase",
   [VersionGroup 1].AddVersion "TAXABLES","BestCase",
```

```
   [VersionGroup 1].Name = "VersionGroup 1"
   [VersionGroup 1].Share = $Unprotected
   ' Declare a variable for a VersionGroups collection.
   Messagebox "Use the VersionGroups method to get versiongroups properties."
   Dim returnVersionGroups As versiongroups
   Set returnVersionGroups = Currentdocument.VersionGroups
   ' Declare a variable to hold the number of versiongroups in the document.
   Dim versionGroupsCount As Long
   ' Get the value of the Count property.
   versionGroupsCount = returnVersionGroups.Count
   MessageBox "The number of versiongroups = + Cstr(versionGroupsCount)
End Sub
```

1-2-3: VersionGroup method

{button ,AL(`H_123_Document_Class;',0)} See list of classes

{button ,AL(`H_123_VERSIONGROUP_METHOD_EXSCRIPT ',1)} See example

Returns a VersionGroup object for the version group matching the specified criteria.

Syntax

versiongroup = *document*.**VersionGroup**(*versiongroupname*, [*lastmodifier*])

Parameters

versiongroupname

String. The name of the version group for the specified file.

lastmodifier

(Optional) String. The name of the user who last modified the version group. This parameter allows you to distinguish between version groups that have the same name but were modified by different users.

Return values

The VersionGroup object corresponding to *versiongroupname*. If there are multiple version groups that use the same name and were last modified by the same user, 1-2-3 chooses a version group to return.

{button ,AL(`H_123_NEWVERSIONGROUP_METHOD_MEMDEF;H_123_VERSIONS_METHOD_MEMDEF',0)} See related topics

```
' Example: VersionGroup and VersionGroups methods
Sub VersionDemo2
   ' Set up two ranges of data, each supporting three versions.
   Messagebox "Set up two ranges of data, each supporting three versions."
   Messagebox "The first range is named BOTTOMLINE."
   [A:A3].Contents = "Item"
   [A:B3].Contents = "Amount"
   [A:A3..A:B3].Font.Bold = True
   [A:A4].Contents = "Income"
   [A:A5].Contents = "Expenses"
   [A:A6].Contents = "Result"
   [A:B4].Contents = "45000"
   [A:B5].Contents = "42000"
   [A:B6].Contents = "+B4-+B5"
   [A:B4..A:B6].Format "US Dollar",0
   ' Create the range name BOTTOMLINE and three versions.
   CurrentDocument.CreateRangeName "BOTTOMLINE",[A:A3..A:B6]
   [BOTTOMLINE].NewVersion "BestCase"
   [BOTTOMLINE.BestCase].Description = "This is the best scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "40000"          ' New data for the version.
   [BOTTOMLINE.Original].MakeCurrent
   [BOTTOMLINE].NewVersion "WorstCase"
   [BOTTOMLINE.WorstCase].Description = "This is the worst case scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "47000"          ' New data for the version.
   ' Set up the second range of data and create the versions.
   Messagebox "The second range is named TAXABLES."
   [A:A9].Contents = "Item"
   [A:B9].Contents = "Amount"
   [A:A9..A:B9].Font.Bold = True
   [A:A10].Contents = "Income"
   [A:A11].Contents = "Returement Account"
   [A:A12].Contents = "Taxable"
   [A:B10].Contents = "45000"
   [A:B11].Contents = "1500"
   [A:B12].Contents = "+B10-+B11"
   [A:B10..A:B12].Format "US Dollar",0
   ' Create the range name TAXABLES and three versions.
   CurrentDocument.CreateRangeName "TAXABLES",[A:A9..A:B12]
   [TAXABLES].NewVersion "BestCase"
   [TAXABLES.BestCase].Description = "This is the best scenario."
   [TAXABLES].VersionBorderVisible = True
   [A:B11].Contents = "2500"          ' New data for the version.
   [TAXABLES.Original].MakeCurrent
   [TAXABLES].NewVersion "WorstCase"
   [TAXABLES.WorstCase].Description = "This is the worst case scenario."
   [TAXABLES].VersionBorderVisible = True
   [A:B11].Contents = "500"           ' New data for the version.
   ' Create a version group named "VersionGroup 1".
   Messagebox "Create a version group named VersionGroup 1."
   CurrentDocument.NewVersionGroup "VersionGroup 1"
   [VersionGroup 1].Description = "Here is a version group."
   ' Add the version BOTTOMLINE.BestCase to the group.
   ' Add the version TAXABLES.BestCase to the group.
   Messagebox "Add versions from BOTTOMLINE and TAXABLES to the group."
   [VersionGroup 1].AddVersion "BOTTOMLINE","BestCase",
   [VersionGroup 1].AddVersion "TAXABLES","BestCase",
```

```
   [VersionGroup 1].Name = "VersionGroup 1"
   [VersionGroup 1].Share = $Unprotected
   ' Declare a variable for a VersionGroups collection.
   Messagebox "Use the VersionGroups method to get versiongroups properties."
   Dim returnVersionGroups As versiongroups
   Set returnVersionGroups = Currentdocument.VersionGroups
   ' Declare a variable to hold the number of versiongroups in the document.
   Dim versionGroupsCount As Long
   ' Get the value of the Count property.
   versionGroupsCount = returnVersionGroups.Count
   MessageBox "The number of versiongroups = " + Cstr(versionGroupsCount)
End Sub
```

1-2-3: Versions method

{button ,AL(`H_123_Range_Class;H_123_VersionGroup_Class;',0)} See list of classes

{button ,AL(`H_123_VERSIONS_METHOD_EXSCRIPT ',1)} See example

Returns a collection of versions matching the specified criteria.

Syntax

versions = *range*.**Versions**([*versionnames]*, *[lastmodifier*])

Parameters

versionnames

(Optional) String. The names of the versions to match. By default, all versions are included.

Note Version names are case-sensitive.

lastmodifier

(Optional) String. The name of a user who last modified one or more versions. If a version was last modified by the user specified in *lastmodifier,* the Versions method adds that version to the collection.

Return values

Returns a Versions object.

{button ,AL(`H_123_DELETEVERSION_METHOD_MEMDEF;H_123_NEWVERSION_METHOD_MEMDEF;H_123_ADDVERSION_METHOD_MEMDEF;H_123_REPORTVERSION_METHOD_MEMDEF',0)} See related topics

```
' Example: ReportVersion, Version, and Versions methods
Sub VersionDemo1
   ' Set up a range of data to support three versions.
   MessageBox "Set up a range of data to support three versions."
   [A:A3].Contents = "Item"
   [A:B3].Contents = "Amount"
   [A:A3..A:B3].Font.Bold = True
   [A:A4].Contents = "Income"
   [A:A5].Contents = "Expenses"
   [A:A6].Contents = "Result"
   [A:B4].Contents = "45000"
   [A:B5].Contents = "42000"
   [A:B6].Contents = "+B4-+B5"
   [A:B4..A:B6].Format "US Dollar",0
   CurrentDocument.CreateRangeName "BOTTOMLINE",[A:A3..A:B6]
   [BOTTOMLINE].NewVersion "BestCase"
   [BOTTOMLINE.BestCase].Description = "This is the best scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "40000"
   [BOTTOMLINE.Original].MakeCurrent
   [BOTTOMLINE].NewVersion "WorstCase"
   [BOTTOMLINE.WorstCase].Description = "This is the worst case scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "47000"
   ' Use the Version method to access properties for the new versions.
   MessageBox "Use the Version method to access properties for the new versions."
   ' Declare a variable as a Version object.
   Dim returnRangeVersion As Version
   ' Select the range containing the versions.
   [BOTTOMLINE].Goto
   [BOTTOMLINE].Select
   ' Assign the variable to one of the versions in the range.
   Set returnRangeVersion = Selection.Version("BestCase")
   ' Declare a variable to hold the version ID of the selected version.
   Dim rangeVersionVersionID As Long
   ' Get the VersionID of the version.
   rangeVersionVersionID = returnRangeVersion.VersionID
   Print "VersionID for version BestCase = " rangeVersionVersionID
   ' Use the Versions method to access in a Versions collection.
   MessageBox "Use the Versions method to get collection properties."
   ' Declare a variable as a Versions collection object.
   Dim returnRangeVersions As versions
   ' Select the range containing the versions.
   [BOTTOMLINE].Goto
   [BOTTOMLINE].Select
   ' Assign the variable to collection of versions in the selected range.
   Set returnRangeVersions = Selection.Versions
   ' Declare a variable to hold the value of the Count property.
   Dim rangeVersionsCount As Long
   ' Get the number of versions in the selected range.
   rangeVersionsCount = Selection.Versions.Count
   Print "Number of versions in range BOTTOMLINE = " rangeVersionsCount
   ' Generate a report about versions in the selected range.
   MessageBox "Generate a report about versions in the selected range."
```

```
   [BOTTOMLINE].ReportVersion "BestCase;Original;",[A:A20],True,True,$Column
End Sub
```

1-2-3: Version method

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_VERSION_METHOD_EXSCRIPT ',1)} See example

Returns a Version object for the version matching the criteria.

Syntax

version = *range*.**Version**(*versionname*, [*lastmodifier*])

Parameters

versionname

String. The name of the version for *range*.

Note Version names are case-sensitive.

lastmodifier

(Optional) String. The name of the user who last modified the version. This parameter allows you to distinguish between two versions that have the same name but were modified by different users.

Return value

A Version object corresponding to *versionname.* If there are multiple versions that use the same name and were last modified by the same user, 1-2-3 chooses a version to return.

{button ,AL(`H_123_ADDVERSION_METHOD_MEMDEF;H_123_REPORTVERSION_METHOD_MEMDEF;H_123_DELETEVERSION_METHOD_MEMDEF;H_123_NEWVERSION_METHOD_MEMDEF;',0)} See related topics

```
' Example: ReportVersion, Version, and Versions methods
Sub VersionDemo1
   ' Set up a range of data to support three versions.
   MessageBox "Set up a range of data to support three versions."
   [A:A3].Contents = "Item"
   [A:B3].Contents = "Amount"
   [A:A3..A:B3].Font.Bold = True
   [A:A4].Contents = "Income"
   [A:A5].Contents = "Expenses"
   [A:A6].Contents = "Result"
   [A:B4].Contents = "45000"
   [A:B5].Contents = "42000"
   [A:B6].Contents = "+B4-+B5"
   [A:B4..A:B6].Format "US Dollar",0
   CurrentDocument.CreateRangeName "BOTTOMLINE",[A:A3..A:B6]
   [BOTTOMLINE].NewVersion "BestCase"
   [BOTTOMLINE.BestCase].Description = "This is the best scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "40000"
   [BOTTOMLINE.Original].MakeCurrent
   [BOTTOMLINE].NewVersion "WorstCase"
   [BOTTOMLINE.WorstCase].Description = "This is the worst case scenario."
   [BOTTOMLINE].VersionBorderVisible = True
   [A:B5].Contents = "47000"
   ' Use the Version method to access properties for the new versions.
   MessageBox "Use the Version method to access properties for the new versions."
   ' Declare a variable as a Version object.
   Dim returnRangeVersion As Version
   ' Select the range containing the versions.
   [BOTTOMLINE].Goto
   [BOTTOMLINE].Select
   ' Assign the variable to one of the versions in the range.
   Set returnRangeVersion = Selection.Version("BestCase")
   ' Declare a variable to hold the version ID of the selected version.
   Dim rangeVersionVersionID As Long
   ' Get the VersionID of the version.
   rangeVersionVersionID = returnRangeVersion.VersionID
   Print "VersionID for version BestCase = " rangeVersionVersionID
   ' Use the Versions method to access in a Versions collection.
   MessageBox "Use the Versions method to get collection properties."
   ' Declare a variable as a Versions collection object.
   Dim returnRangeVersions As versions
   ' Select the range containing the versions.
   [BOTTOMLINE].Goto
   [BOTTOMLINE].Select
   ' Assign the variable to collection of versions in the selected range.
   Set returnRangeVersions = Selection.Versions
   ' Declare a variable to hold the value of the Count property.
   Dim rangeVersionsCount As Long
   ' Get the number of versions in the selected range.
   rangeVersionsCount = Selection.Versions.Count
   Print "Number of versions in range BOTTOMLINE = " rangeVersionsCount
   ' Generate a report about versions in the selected range.
   MessageBox "Generate a report about versions in the selected range."
```

```
   [BOTTOMLINE].ReportVersion "BestCase;Original;",[A:A20],True,True,$Column
End Sub
```

1-2-3: WhatIfTable1 method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Substitutes values for one variable in one or more formulas and enters the results in an output range.

Syntax

document.**WhatIfTable1** *outputrange, inputcell*

Parameters

outputrange

Variant. The name or address of a range that contains the formulas, a list of input values that the formula uses in place of the variable, and blank cells where 1-2-3 places the results.

inputcell

Variant. The name or address of the cell in which 1-2-3 temporarily enters values for the variable while performing the calculations required to create the what-if table.

Inputcell1 must be outside of *outputrange*.

Return values

None

1-2-3: WhatIfTable2 method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Substitutes values for two variables in one formula and enters the results in an output range.

Syntax

document.**WhatIfTable2** *outputrange, inputcell1, inputcell2*

Parameters

outputrange

Variant. The name or address of a range that contains the formula, a list of input values that the formula uses in place of the variables, and blank cells where 1-2-3 places the results.

inputcell1, *inputcell2*

Variant. The names or addresses of the first and second cells in which 1-2-3 temporarily enters values while performing the calculations required to create the table.

Inputcell1 and *inputcell2* must be outside of *outputrange*.

Return values

None

1-2-3: WhatIfTable3 method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Substitutes values for three variables in one formula and enters the results in an output range.

Syntax

document.**WhatIfTable3** *outputrange, inputcell1, inputcell2, inputcell3, formula*

Parameters

outputrange

Variant. The name or address of a range that a list of input values that the formula uses in place of the variables, and blank cells where 1-2-3 places the results.

inputcell1, *inputcell2, inputcell3*

Variant. The names or addresses of the first, second, and third cells in which 1-2-3 temporarily enters values while performing the calculations required to create the table.

Inputcell1, inputcell2, and *inputcell3* must be outside of *outputrange*.

formula

Variant. The name or address of a cell containing the formula that has the three variables you want to change.

Return Values

None

1-2-3: WhatIfTableReset method

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

Clears the output ranges and input cells for all what-if tables in the current file.

Syntax

document.**WhatIfTableReset**

Parameters

None

Return values

None

1-2-3: ZoomIn method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_ZOOMIN_METHOD_EXSCRIPT ',1)} See example

Increases the current zoom value for all sheets in the document by 25% to 400%.

Syntax

document.**ZoomIn**

Parameters

None

Return values

None

{button ,AL(`H_123_ZOOMOUT_METHOD_MEMDEF;H_123_ZOOMRESET_METHOD_MEMDEF;H_123_ZOOMTO_METHOD_MEMDEF',0)} See related topics

```
' Example: ZoomIn, ZoomOut, ZoomReset, and ZoomTo methods
Sub ZoomScales
   ' Zoom out to a preset scale.
   Msgbox "Zoom out to a preset scale."
   CurrentDocument.ZoomOut

   ' Zoom in to the next preset scale.
   Msgbox "Zoom in to the next preset scale."
   CurrentDocument.ZoomIn

   ' Zoom in to the next preset scale.
   Msgbox "Zoom in to the next preset scale."
   CurrentDocument.ZoomIn

   ' Reset the Zoom factor to the current custom zoom factor.
   Msgbox "Reset the Zoom factor to the default."
   CurrentDocument.ZoomReset

   ' Zoom out to 25%.
   Msgbox "Zoom out to 25%."
   CurrentDocument.ZoomTo 25

   ' Zoom in to 200%.
   Msgbox "Zoom in to 200%."
   CurrentDocument.ZoomTo 200

   ' Set the custom zoom factor to 80% and zoom to 80%.
   Msgbox "Set the ZoomScale property to 80%."
   CurrentDocument.ZoomScale = 80
   CurrentDocument.ZoomReset
End Sub
```

1-2-3: ZoomMapIn method

{button ,AL(`H_123_MAP_CLASS;',0)} See list of classes

Increases the display size of the map by 100%.

Syntax

map.**ZoomMapIn**

Parameters

None

Return values

None

{button ,AL(`H_123_RECENTERMAP_METHOD_MEMDEF;H_123_REDRAWMAP_METHOD_MEMDEF;H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_ZOOMMAPIN_METHOD_MEMDEF;H_ZOOMMAPOUT_METHOD_MEMDEF;H_ZOOMMAPRESET_METHOD_MEMDEF;H_ZOOMMAPTO_METHOD_MEMDEF;H_ZOOMMAPTORECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: ZoomMapOut method

{button ,AL(`H_123_MAP_CLASS;',0)} See list of classes

Shrinks the display size of the map by 50%.

Syntax

map.**ZoomMapOut**

Parameters

None

Return values

None

{button ,AL(`H_123_RECENTERMAP_METHOD_MEMDEF;H_123_REDRAWMAP_METHOD_MEMDEF;H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_ZOOMMAPIN_METHOD_MEMDEF;H_ZOOMMAPOUT_METHOD_MEMDEF;H_ZOOMMAPRESET_METHOD_MEMDEF;H_ZOOMMAPTO_METHOD_MEMDEF;H_ZOOMMAPTORECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: ZoomMapReset method

{button ,AL(`H_123_MAP_CLASS;',0)} See list of classes

Displays the map as it was when it was created.

Syntax

map.**ZoomMapReset**

Parameters

None

Return values

None

{button ,AL(`H_123_RECENTERMAP_METHOD_MEMDEF;H_123_REDRAWMAP_METHOD_MEMDEF;H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_ZOOMMAPIN_METHOD_MEMDEF;H_ZOOMMAPOUT_METHOD_MEMDEF;H_ZOOMMAPRESET_METHOD_MEMDEF;H_ZOOMMAPTO_METHOD_MEMDEF;H_ZOOMMAPTORECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: ZoomMapTo method

{button ,AL(`H_123_MAP_CLASS;',0)} See list of classes

Sets the view scale for the map to the specified percentage.

Syntax

map.**ZoomMapTo**(*viewscale*)

Parameters

viewscale

Long. Any long from 0 (zero) - 10000. The percentage to use for the map's view scale.

Return values

None

{button ,AL(`H_123_RECENTERMAP_METHOD_MEMDEF;H_123_REDRAWMAP_METHOD_MEMDEF;H_123_REMOVEOVERLAY_METHOD_MEMDEF;H_123_ZOOMMAPIN_METHOD_MEMDEF;H_123_ZOOMMAPOUT_METHOD_MEMDEF;H_123_ZOOMMAPRESET_METHOD_MEMDEF;H_123_ZOOMMAPTO_METHOD_MEMDEF;H_123_ZOOMMAPTORECTANGLE_METHOD_MEMDEF',0)} See related topics

1-2-3: ZoomOut method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_ZOOMOUT_METHOD_EXSCRIPT ',1)} See example

Decreases the current zoom value for all sheets in the document by by a value between 25% and 400%.

Syntax

document.**ZoomOut**

Parameters

None

Return values

None

{button ,AL(`H_123_ZOOMIN_METHOD_MEMDEF;H_123_ZOOMRESET_METHOD_MEMDEF;H_123_ZOOMTO_METHOD_MEMDEF',0)} See related topics

```
' Example: ZoomIn, ZoomOut, ZoomReset, and ZoomTo methods
Sub ZoomScales
   ' Zoom out to a preset scale.
   Msgbox "Zoom out to a preset scale."
   CurrentDocument.ZoomOut

   ' Zoom in to the next preset scale.
   Msgbox "Zoom in to the next preset scale."
   CurrentDocument.ZoomIn

   ' Zoom in to the next preset scale.
   Msgbox "Zoom in to the next preset scale."
   CurrentDocument.ZoomIn

   ' Reset the Zoom factor to the current custom zoom factor.
   Msgbox "Reset the Zoom factor to the default."
   CurrentDocument.ZoomReset

   ' Zoom out to 25%.
   Msgbox "Zoom out to 25%."
   CurrentDocument.ZoomTo 25

   ' Zoom in to 200%.
   Msgbox "Zoom in to 200%."
   CurrentDocument.ZoomTo 200

   ' Set the custom zoom factor to 80% and zoom to 80%.
   Msgbox "Set the ZoomScale property to 80%."
   CurrentDocument.ZoomScale = 80
   CurrentDocument.ZoomReset
End Sub
```

1-2-3: ZoomReset method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_ZOOMRESET_METHOD_EXSCRIPT',1)} See example

Restores the view scale for the file to the custom scale setting.

Syntax

document.**ZoomReset**

Parameters

None

Return values

None

{button ,AL(`H_123_ZOOMIN_METHOD_MEMDEF;H_123_ZOOMOUT_METHOD_MEMDEF;H_123_ZOOMTO_METHOD_MEMDEF',0)} See related topics

```
' Example: ZoomIn, ZoomOut, ZoomReset, and ZoomTo methods
Sub ZoomScales
   ' Zoom out to a preset scale.
   Msgbox "Zoom out to a preset scale."
   CurrentDocument.ZoomOut

   ' Zoom in to the next preset scale.
   Msgbox "Zoom in to the next preset scale."
   CurrentDocument.ZoomIn

   ' Zoom in to the next preset scale.
   Msgbox "Zoom in to the next preset scale."
   CurrentDocument.ZoomIn

   ' Reset the Zoom factor to the current custom zoom factor.
   Msgbox "Reset the Zoom factor to the default."
   CurrentDocument.ZoomReset

   ' Zoom out to 25%.
   Msgbox "Zoom out to 25%."
   CurrentDocument.ZoomTo 25

   ' Zoom in to 200%.
   Msgbox "Zoom in to 200%."
   CurrentDocument.ZoomTo 200

   ' Set the custom zoom factor to 80% and zoom to 80%.
   Msgbox "Set the ZoomScale property to 80%."
   CurrentDocument.ZoomScale = 80
   CurrentDocument.ZoomReset
End Sub
```

1-2-3: ZoomTo method

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_ZOOM_PROPERTY_EXSCRIPT;H_123_ZOOMTO_METHOD_EXSCRIPT;',1)} See example

Sets the view scale for all sheets in the file to the specified magnification percentage. Increasing the view scale increases magnification; decreasing it reduces magnification.

Syntax

document.**ZoomTo** *viewscale*

Parameters

viewscale

Long. Any long from 25 - 400. The magnification factor of the view scale expressed as a percent. The default view scale for new files can be set using the View menu Zoom commands.

Return values

None

{button ,AL(`H_123_ZOOMIN_METHOD_MEMDEF;H_123_ZOOMOUT_METHOD_MEMDEF;H_123_ZOOMRESET_METHOD_MEMDEF;',0)} See related topics

```
'Example: ZoomTo method
'Toggle display size of cells between 75% and 100%
   'Check current zoom percentage
   If .Zoom = 75 Then
      'Zoom in or out, accordingly
      .ZoomTo 100
   Else
      .ZoomTo 75
   End If
```

1-2-3: ActiveCell property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ACTIVECELL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns a Range object containing just the active cell.

Data type

Range

Syntax

Set *value* **=** *sheet*.**ActiveCell**

Legal values

The legal value for the ActiveCell property is a single-cell Range object.

```
' Example: ActiveCell property and SetActiveCell method
' To verify the active cell.
[C1..D5].Select
[D3].SetActiveCell
'Verify that the active cell is set and returned correctly.
Msgbox [A].ActiveCell.Name
```

```
' Example: ActiveDocument property
' To read the name of the active document.
Msgbox "The active document is " + CurrentApplication.ActiveDocument.Name
```

1-2-3: ActiveDocument property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_ACTIVEDOCUMENT_EXSCRIPT',1)} See example

(Read-only) Returns the current document object.

Data type

Document

Syntax

Set *value* **=** *application*.**ActiveDocument**

Legal values

The value of the ActiveDocument property is a Document object.

1-2-3: ActiveDocWindow property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the active document window.

Data type

DocWindow

Syntax

Set *application*.**ActiveDocWindow =** *value*

Set *value* **=** *application*.**ActiveDocWindow**

Legal values

The value for the ActiveDocWindow property can be any valid document window.

1-2-3: Active property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_ACTIVE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determines whether the object you specify is the active object.

Data type

Variant (Boolean)

Syntax

value **=** *object***.Active**

Legal values

Value	Description
TRUE	The specified object is the active object.
FALSE	The specified object is not the active object.

```
' Example: Active property
' To determine whether a file is active.
Forall win In CurrentDocument.DocWindows
    If win.Active = False Then
       Msgbox win.Name + "is not active."
    Else
       Msgbox win.Name + "is active."
    End If
End Forall
```

1-2-3: Addins property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns a collection of add-in names that are currently registered.

Data type

Strings

Syntax

Set *strings* **=** *application*.**Addins**

Legal values

The value of the Addins property is the names of add-ins that are currently registered.

Usage

An add-in must be registered in order to be loaded.

{button ,AL(`H_123_ISADDINLOADED_METHOD_MEMDEF',0)} See related topics

1-2-3: AlignOverColumns property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ALIGNOVERCOLUMNS_PROPERTY_EXSCRIPT',1)} See example

Determines whether the text in the leftmost cell is aligned over the columns within the range. The text alignment for the range is set by the TextHorizontalAlign property.

Data type

Variant (Boolean)

Syntax

range.**AlignOverColumns =** *value*

value **=** *range*.**AlignOverColumns**

Legal values

Value	Description
TRUE	Align text over the columns.
FALSE	Do not align text over the columns.

```
' Example: AlignOverColumns, Contents, and TextHorizontalAlign properties (Text Only)
' Select a range and align the text over columns.
Sub AlignText
    [A:B5].Contents = "Test"
    [A:B5].TextHorizontalAlign = $AlignCenter
    [A:B5..A:C5].Select
    [A:B5..A:C5].AlignOverColumns = True
End Sub
```

1-2-3: AllNames property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ALLNAMES_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the address of the current range and all range names for that range.

Data type

Strings

Syntax

value **=** *range***.AllNames**

Legal values

The values of the AllNames property are all range names for the specified range, all associated version names, and the range address. The AllNames property always returns the range address, at least.

```
' Example: AllNames property and CreateRangeName method
' Create 3 different range names on the same range.
' Print out all names for the range.
Sub PrintNames
    CurrentDocument.CreateRangeName "xyz1",[A:B10..A:E15]
    CurrentDocument.CreateRangeName "eieio",[A:B10..A:E15]
    CurrentDocument.CreateRangeName "b10toe15",[A:B10..A:E15]

    Forall x In [A:B10..A:E15].AllNames
        Print x
    End Forall
End Sub

'The output will be:
'   A:B10..A:E15
'   B10TOE15
'   EIEIO
'   XYZ1
```

1-2-3: AllowsUpdates property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Determines whether the user can update the data retrieved for a query, using the Query Table - Update command. Setting this property also causes the data to be refreshed.

Data type

Variant (Boolean)

Syntax

*dataquery***.AllowsUpdates =** *value*

value **=** *dataquery***.AllowsUpdates**

Legal values

Value	Description
TRUE	The query table will accept updates.
FALSE	(Default) The query table will not accept updates.

Usage

This property can only be set if there is only one source table associated with the query. It is not valid if there are joined tables, or if the query table has an aggregate, a computed column, sorted fields, or if the unique setting is on, or if records-limited is on.

1-2-3: AllPagesPrint property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether to print all of the pages in the print selection.

Data type

Variant (Boolean)

Syntax

printsettings.**AllPagesPrint =** *value*

value **=** *printsettings*.**AllPagesPrint**

Legal values

Value	Description
TRUE	(Default) Print all pages in the print selection.
FALSE	Print only the pages specified by the PageFrom and PageTo properties.

1-2-3: AlwaysReserve property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_ALWAYSRESERVE_PROPERTY_EXSCRIPT',1)} See example

Determines whether the file opens as read-write (reserved) or read-only.

Data type

Variant (Boolean)

Syntax

value **=** *document***.AlwaysReserve**

*document***.AlwaysReserve =** *value*

Legal values

Value	Description
TRUE	(Default) The file will open as read-write.
FALSE	The file will open as read-only.

```
' Example: AlwaysReserve property
' Reserve the file.
CurrentDocument.AlwaysReserve = True
```

1-2-3: Anchor property

{button ,AL(`;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

Sets or returns how an object is fastened to the cells behind it.

Data type

Variant (AnchorType enumeration)

Syntax

object.**Anchor =** *value*

value **=** *object*.**Anchor**

Legal values

Value	Description
$TopLeftAndBottomRight	You can move and size the object when you move, size, and hide cells behind it.
$TopLeft	You can move, but not size, the object when you move, size, and hide cells behind it.
$NotFastened	You can detach the object from the underlying cells. Then you can move, size, or hide cells behind the selected object without moving or resizing the object.

Usage

Anchoring applies to the following objects: Arc, ButtonControl, Chart, DrawLine, DrawObject, EditText, Ellipse, Freehand, Group, Legend, Map, MapTitle, OLEObject, Picture, MapPlot, Polygon, Polyline, Rectangle, RoundedRectangle.

1-2-3: ApplicationMaximized property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_APPLICATIONMAXIMIZED_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display 1-2-3 in a maximized window when starting.

Data type

Variant (Boolean)

Syntax

application.**ApplicationMaximized =** *value*

value **=** *application*.**ApplicationMaximized**

Legal values

Value	Description
TRUE	Display 1-2-3 in a maximized window.
FALSE	(Default) Do not display 1-2-3 in a maximized window.

```
' Example: ApplicationMaximized property
' Set the application to display as maximized when opened.
CurrentApplication.ApplicationMaximized = True
```

1-2-3: ApplicationWindow property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_CASCADE_METHOD_EXSCRIPT;H_123_APPLICATIONWINDOW_PROPERTY_EXSCRIPT',1) } See example

(Read-only) Returns the application window for the current application.

Data type

ApplicationWindow

Syntax

Set *applicationwindow* **=** *application*.**ApplicationWindow**

Legal Values

The value of the ApplicationWindow property is the ApplicationWindow object.

```
' Example: ApplicationWindow property and Cascade method
' Create two document windows and cascade them.
Dim appwindow As ApplicationWindow
Dim doc1 As Document
Dim doc2 As Document
Set appwindow = CurrentApplication.ApplicationWindow
Set doc1 = CurrentApplication.NewDocument("Document #1")
Set doc2 = CurrentApplication.NewDocument("Document #2")
' Cascade the document windows.
appwindow.Cascade
```

1-2-3: Application property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_ARC_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_APPLICATION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the application object. There can be only one application object per running application, that is, one application per 1-2-3 executable running. There is a single window associated with each application.

Data type

Application

Syntax

Set *application* **=** *object***.Application**

Legal values

The legal value for the Application property is the application object.

```
' Example: Application property
' The Application property is best illustrated in the context of OLE automation.
' Assume this script is being run from any Lotus SmartSuite product
' or from VisualBasic.
Dim docvar As Variant
Dim appvar As Variant
Set docvar = CreateObject("Lotus123.Workbook.97")
' Get a reference to the 1-2-3 application object.
Set appvar = docvar.Application
Msgbox "The application name is "+appvar.name
```

1-2-3: ArgumentSeparator property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_ARGUMENTSEPARATOR_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the argument separator that is set in the Windows Control Panel. For more information, see your Microsoft documentation.

Data type

String

Syntax

value **=** *application*.**ArgumentSeparator**

Legal values

The value of the ArgumentSeparator property is the list separator set in the Windows Control Panel.

```
' Example: ArgumentSeparator property
' To return the argument separator.
Msgbox "The regional settings list separator is " + _
        CurrentApplication.ArgumentSeparator
```

1-2-3: Arrow property

{button ,AL(`H_123_ARC_CLASS;H_123_DRAWLINE_CLASS;H_123_FREEHAND_CLASS;H_123_POLYLINE_CLASS;H_123_DRAWCOLLECTION_CLASS',0)} See list of classes

Sets or returns the type of arrowhead that is added to an object.

Data type

Variant (ArrowType enumeration)

Syntax

object.**Arrow =** *value*

value **=** *object*.**Arrow**

Legal values

Value	Description
$Head	Add an arrowhead to the beginning of the object.
$Tail	Add an arrowhead to the end of the object.
$Both	Add an arrowhead to both ends of the object.
$None	Do not add any arrowheads to the object.

1-2-3: Authors property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_AUTHORS_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns a string collection containing the name of the person who created the file and the name of the last person to modify that file.

Data type

Strings

Syntax

Set *strings* **=** *document***.Authors**

Legal values

The value of the Authors property is the Strings object containing the names of the persons who created and last modified the file.

```
' Example: Authors property
' To return the author and last editor of the document.
Msgbox "The original author of this document is " + _
        CurrentDocument.Authors(0)
Msgbox "The last editor of this document is " + CurrentDocument.Authors(1)
```

1-2-3: Author property

{button ,AL(`;H_123_DOCUMENT_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS',0)} See list of classes

{button ,AL(`H_123_AUTHOR_PROPERTY_EXSCRIPT;H_123_KEYWORDS_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns the name of the person who created the workbook, version, or version group.

Data type

String

Syntax

value **=** *object***.Author**

Legal values

The value of the Author property is the string containing the name of the person who created the object.

```
' Example: Author property
' To return the author of the document.
Msgbox "The author of this document is " + CurrentDocument.Author
```

1-2-3: AutoExecMacrosEnabled property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_AUTOEXECMACROSENABLED_PROPERTY_EXSCRIPT',1)} See example

Determines whether autoexecute macros run automatically when you open a workbook.

Data type

Variant (Boolean)

Syntax

application.**AutoExecMacrosEnabled =** *value*

value **=** *application*.**AutoExecMacrosEnabled**

Legal values

Value	Description
TRUE	(Default) Run the autoexecute macros.
FALSE	Do not run the autoexecute macros.

```
' Example: AutoexecMacrosEnabled property
' Set autoexecute macros to run automatically.
CurrentApplication.AutoExecMacrosEnabled = True
```

1-2-3: AutoOpenPath property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_AUTOOPENPATH_PROPERTY_EXSCRIPT',1)} See example

Sets or returns a folder. The files in this folder are automatically opened when the application starts.

Data type

String

Syntax

application.**AutoOpenPath** = *value*

value **=** *application*.**AutoOpenPath**

Legal values

The value of the AutoOpenPath property is the name of the folder you specify in the File - User Setup - 1-2-3 Preferences dialog box, under Automatically opened files.

```
' Example: AutoOpenPath property
' To return the directory of the documents that are opened automatically.
Msgbox "The documents in the directory " + _
        CurrentApplication.AutoOpenPath + _
        are opened automatically when 1-2-3 starts."
```

1-2-3: AutoRedraw property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS',0)} See list of classes

Determines whether maps are set to redraw automatically when data changes.

Data type

Variant (Boolean)

Syntax

object.**AutoRedraw =** *value*

value **=** *object*.**AutoRedraw**

Legal values

Value	Description
TRUE	(Default) Maps redraw automatically.
FALSE	Maps do not redraw automatically.

Usage

If the value of the AutoRedraw property is false, 1-2-3 redraws maps only when it reaches a RedrawMap method in a script or when the user chooses the Map - Redraw command.

1-2-3: AutoRefresh property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Determines whether to automatically refresh a query table whenever a change is made to the query that would affect the results.

The following methods and properties will always refresh the table automatically: AddSelectField, RemoveSelectField, setting SelectFields, setting BaseSourceTable, CreateComputedField, DeleteComputedField, Join, and setting AllowsUpdates to TRUE.

Data type

Variant (Boolean)

Syntax

*dataquery***.AutoRefresh =** *value*

value **=** *dataquery***.AutoRefresh**

Legal values

Value	Description
TRUE	(Default) Refresh the query table automatically.
FALSE	Do not refresh the query table automatically.

Usage

If the value of the AutoRefresh property is false, 1-2-3 refreshes query tables only when it reaches a RefreshQuery method in a script or when the user chooses the Query Table - Refrsh command.

1-2-3: AutoUpdate property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

Determines whether to automatically update the linked object when the source changes.

Data type

Variant (Boolean)

Syntax

object.**AutoUpdate =** *value*

value **=** *object*.**AutoUpdate**

Legal values

Value	Description
TRUE	(Default) Update the linked object automatically.
FALSE	Do not update the linked object automatically.

Usage

If the value of the AutoUpdate property is false, 1-2-3 updates links only when it reaches an Update method in a script or when the user chooses Edit - Manage Links and clicks the Update Now button in the Manage Links dialog box.

1-2-3: AvailableMemory property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the amount of available memory in bytes.

Data type

Long

Syntax

value **=** *application*.**AvailableMemory**

Legal Values

The value for the AvailableMemory property is the amount of available memory.

1-2-3: BackColor property

{button ,AL(`H_123_BACKGROUND_CLASS',0)} See list of classes

{button ,AL(`H_123_BACKCOLOR_PROPERTY_EXSCRIPT;H_123_SHOWDRAWLAYER_PROPERTY_EXSCRIPT;H_123_GREEN_PROPERTY_EXSCRIPT;H_123_ISHIDDEN_AND_ISPROTECTED_PROPERTIES_EXSCRIPT',1)} See example

Sets or returns the background color for an object.

Data type

Color

Syntax

Set *background*.**BackColor =** *color*

Set *color* **=** *background*.**BackColor**

Legal Values

The value of the BackColor property is a Color object.

```
' Example: BackColor, Background, Bold, Colors, Contents, Font, Fontcolor,
' and Pattern properties
' Declare the variables.
    Dim testapp As Application
    Dim testfont As Font
    Dim y As Color
    Dim z As Color
    Dim range1 As Range
    Set testapp = CurrentApplication
' Declare and name a range.
    Set range1 = [A:A2..A:A10]
' Set the variable testfont as the font for the A1 cell and make
' the font bold. Set the variables y and z to the colors red and white.
    Set testfont = [A1].Font
    testfont.Bold = True
    Set y = testapp.Colors("White")
    Set z = testapp.Colors("Red")
' Set the colors for the first column and the background pattern for
' the named range. Set the contents of cell A1 to read "label here".
    Set testfont.Fontcolor = y
    Set range1.Background.Backcolor = z
    range1.Background.Pattern = $runningbricks
    [A1].Contents = "label here"
```

1-2-3: Background property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_CHART_CLASS;;H_123_DRAWCOLLECTION_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_BACKCOLOR_PROPERTY_EXSCRIPT;H_123_SHOWDRAWLAYER_PROPERTY_EXSCRIPT;H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;H_123_BACKGROUND_PROPERTY_EXSCRIPT;H_123_PATTERN_PROPERTY_EXSCRIPT;H_123_GREEN_PROPERTY_EXSCRIPT;H_123_ISHIDDEN_AND_ISPROTECTED_PROPERTIES_EXSCRIPT',1)} See example

With the Pattern, Color, or BackColor property, sets or returns the pattern, color, or background color for an object.

Data type

Background

Syntax

Set *object*.**Background =** *background*

Set *object*.**Background.BackColor =** *color*

Set *object*.**Background.Color =** *color*

Set *object*.**Background.Pattern =** *pattern*

Set *background* **=** *object*.**Background**

Set *color* **=** *object*.**Background.BackColor**

Set *color* **=** *object*.**Background.Color**

Set *pattern* **=** *object*.**Background.Pattern**

Legal values

A Background object.

See the Pattern property for a list of patterns.

See the Color palette for a list of colors.

```
' Example: Background and Pattern properties
' Add a background pattern.
Sub AddPattern
   [A:A5..A:H25].Background.Pattern = $DoubleRightHatch
End Sub
```

1-2-3: BaseMapName property

{button ,AL(`;H_123_DRAWCOLLECTION_CLASS;H_123_MAP_CLASS;H_123_GROUP_CLASS',0)} See list of classes

(Read-only) Returns the file name of a base map. A base map is a map one starts with before adding overlays.

Data type

String

Syntax

value **=** *object***.BaseMapName**

Legal values

The value of the BaseMapName property is a string containing the name of a map. Map names for maps that are available with 1-2-3 include the following:

World Countries

USA by State

Alaska

Hawaii

Canada by Province

European Union by Region

Europe by Country

Japan by Prefecture

Mexico by Estado

Australia by State

Taiwan

1-2-3: BaseSourceTable property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Sets or returns the base (first) source table. If there is a joined table, all other tables are joined to the base table.

Data type

String

Syntax

dataquery.**BaseSourceTable =** *value*

value **=** *dataquery*.**BaseSourceTable**

Legal values

The value of the BaseSourceTable property is a string containing the range name or address of a database table.

Usage

When you set a new table as the base source table, it must have the same number of fields and the same field names as the current base source table.

1-2-3: BeepsOnError property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_BEEPSONERROR_PROPERTY_EXSCRIPT',1)} See example

Determines whether an audible beep sounds when an error occurs.

Data type

Variant (Boolean)

Syntax

application.**BeepsOnError =** *value*

value **=** *application*.**BeepsOnError**

Legal values

Value	Description
TRUE	(Default) Sound a beep when an error occurs.
FALSE	Do not sound a beep when an error occurs.

```
' Example: BeepsOnError property
' Set the property to sound a beep when an error occurs.
CurrentApplication.BeepsOnError = True
```

1-2-3: BinRange property

{button ,AL(`H_123_MAPBINS_CLASS',0)} See list of classes

Sets or returns the range that contains the data 1-2-3 uses to group data into map data bins.

Data type

Range

Syntax

Set *mapbins*.**BinRange =** *value*

Set *value* **=** *mapbins*.**BinRange**

Legal values

The value of the BinRange property is any valid range.

1-2-3: BinsUsed property

{button ,AL(`H_123_MAPBINS_CLASS',0)} See list of classes

(Read-only) Returns the number of map data bins that contain values for the specified collection of bins.

Data type

Long

Syntax

value **=** *mapbins*.**BinsUsed**

Legal values

The value of the BinsUsed property is the number of bins that contain values.

1-2-3: BinType property

{button ,AL(`H_123_MAPBINS_CLASS',0)} See list of classes

Sets or returns the way 1-2-3 groups data into map data bins for a collection of bins: Exact Match or Upper Limit.

Data type

Variant (MapBinType enumeration)

Syntax

mapbins.**BinType =** *value*

value **=** *mapbins*.**BinType**

Legal values

Value	Description
$ExactMatch	Display in the map data that is the same as the bin value only.
$UpperLimit	Display in the map data that is less than or equal to the bin value.

1-2-3: Blue property

{button ,AL(`H_123_COLOR_CLASS',0)} See list of classes

{button ,AL(`H_123_BLUE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the blue component of a color.

Data type

Long

Syntax

value **=** *color***.Blue**

Legal values

The value of the Blue property is an integer from 0 - 255.

```
' Example: Blue and Colors properties
' Return the Blue value of the color.
Dim blueval As Long
Dim mycolor As Color
Set mycolor = CurrentApplication.Colors("magenta")
blueval = mycolor.Blue
Print blueval
```

1-2-3: Bold property

{button ,AL(`H_123_FONT_CLASS',0)} See list of classes

{button ,AL(`H_123_BACKCOLOR_PROPERTY_EXSCRIPT;H_123_STRIKETHROUGH_PROPERTY_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT;H_123_BOLD_PROPERTY_EXSCRIPT',1)} See example

Sets or returns whether the data is styled using the bold attribute.

Data type

Variant (Boolean)

Syntax

font.**Bold =** *value*

value **=** *font*.**Bold**

Legal values

Value	Description
TRUE	Apply the bold attribute to the data.
FALSE	(Default) Do not apply the bold attribute to the data.

```
' Example: Bold, Contents, and Font properties
' Make the title bold.
[A1].Contents = "Quarterly Report"
[A1].Font.Bold = True
```

1-2-3: BottomBorder property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_BOTTOMBORDER_PROPERTY_EXSCRIPT',1)} See example

With the Style or Color property, sets or returns the style or color of the bottom border of a range.

Data type

RangeBorder

Syntax

Set *range*.**BottomBorder.Style =** *value*

Set *range*.**BottomBorder.Color =** *value*

Set *value* **=** *range*.**BottomBorder.Style**

Set *value* **=** *range*.**BottomBorder.Color**

Legal values

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

```
'Example: BottomBorder, Color, and ColorName properties
'Select a range and apply a bottom border style to the range.
[A1..B5].Select
Selection.BottomBorder.Style = $LongDashBorder
Selection.BottomBorder.Color.ColorName = "red"
```

1-2-3: BottomMargin property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the bottom margin print setting in twips.

Data type

Long

Syntax

printsettings.**BottomMargin =** *value*

value **=** *printsettings*.**BottomMargin**

Legal values

The combined top and bottom margin settings cannot be greater than the length of the paper.

1-2-3: CalcIterations property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CALCITERATIONS_PROPERTY_EXSCRIPT',1)} See example

Sets the number of passes 1-2-3 makes to recalculate data.

Data type

Long

Syntax

object.**CalcIterations =** *value*

value **=** *object*.**CalcIterations**

Legal values

The value of the CalcIterations property is from 1 - 50.

```
' Example: CalcIterations property
' Set the recalculation iterations value for a document.
CurrentDocument.CalcIterations = 5
```

1-2-3: CalcMode property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_DESCRIPTION_PROPERTY_EXSCRIPT;H_123_CALCMODE_PROPERTY_EXSCRIPT',1)} See example

Sets or returns whether 1-2-3 recalculates data automatically or manually.

Data type

Variant (RecalcMode enumeration)

Syntax

object.**CalcMode =** *value*

value **=** *object*.**CalcMode**

Legal values

Value	Description
$Automatic	(Default) Automatically recalculate all corresponding formulas when a change has been made to a cell.
$Manual	Recalculate formulas only after the following actions or commands: F9, {Calc}, .Calc, or .RecalcRange.

```
' Example: CalcMode property
' Set the recalculation mode to manual for the document.
CurrentDocument.CalcMode = $Manual
```

1-2-3: CalcOrder property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CALCORDER_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the order in which 1-2-3 recalculates data: on all values, by columns, or by rows.

Data type

Variant (RecalcOrder enumeration)

Syntax

object.**CalcOrder =** *value*

value **=** *object*.**CalcOrder**

Legal values

Value	Description
$Natural	(Default) Recalculate all values on which the formula depends before recalculating the formula.
$Column	Recalculate by column, beginning with A:A1.
$Row	Recalculate by row, beginning with A:A1.

```
' Example: CalcOrder property
' Recalculate values in the document by column.
CurrentDocument.CalcOrder = $Columns
```

1-2-3: Caption property

{button ,AL(`;H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)
} See list of classes

(Read-only) Returns the title of a window.

Data type

String

Syntax

value **=** *object***.Caption**

Legal values

The value for the Caption property is a string containing the window title.

1-2-3: CellCommentsPrint property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether cell comments for the corresponding print range will print. Cell comments will print in a list starting on a new page, following all other data.

Data type

Variant (Boolean)

Syntax

printsettings.**CellCommentsPrint =** *value*

value **=** *printsettings*.**CellCommentsPrint**

Legal values

Value	Description
TRUE	Print cell comments.
FALSE	(Default) Do not print cell comments.

1-2-3: CellComment property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;H_123_CELLCOMMENT_PROPERTY_EXSCRIPT',1)} See example

Sets or returns a comment that is attached to a cell or range, which can be viewed by the Range - Range Properties InfoBox.

Data type

String

Syntax

range.**CellComment =** *value*

value **=** *range*.**CellComment**

Legal values

The maximum length of the string is 1024 characters.

Usage

You can set or return a comment for a single cell only. If you set a cell comment for a range, only the first cell in the range contains the comment.

```
' Example: CellComment property
' Set the comment of a cell.
Dim commentfortotal As String
commentfortotal = "This is the grand total for the year."
[A:B9].CellComment = commentfortotal
```

1-2-3: CellDisplay property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_CELLDISPLAY_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the formatted cell as it is displayed.

Data type

String

Syntax

value **=** *range***.CellDisplay**

Legal Value

The value of the CellDisplay propery is a string containing the formatted cell.

```
' Example: CellDisplay, Contents, FormatDecimals, and FormatName properties
' Enter a value, format the value, and display the formatted value.
Sub FormatVal
    [A:B5].Select
    Selection.Contents = "0.15"
    Selection.FormatName = "Percent"
    Selection.FormatDecimals = 0
    Messagebox Selection.CellDisplay
End Sub
'The messagebox will display:   15%
```

1-2-3: Cells property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_CELLS_PROPERTY_EXSCRIPT;H_123_FONTCOLOR_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns a range collection, with each element of the collection being a single-cell range. Each iteration of the collection goes to the next cell in the range.

Data type

Ranges

Syntax

Set *ranges* **=** *range***.Cells**

Legal values

The value of the Cells property is a range collection of single-cell ranges.

```
' Example: Cells and Contents properties
' Returns a range collection.
'Create a new sheet.
CurrentDocument.NewSheet $After,1,True
'Declare a variable of type Ranges.
Dim rangetoiterate As Ranges
'Use a 3D source range.
Set rangetoiterate = [A:A1..B:C3].Cells
'A variable for use in the Forall.
Dim indx As Integer
'Collections start at zero.
indx=0
'This loop sets the Contents property to its index value in the
'Cells collection.
Forall onecell in rangetoiterate
  'CellValue is the index.
   onecell.Contents = Cstr(indx)
   indx=indx + 1
End Forall
```

1-2-3: CellValue property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_CELLVALUE_PROPERTY_EXSCRIPT;H_123_FONTCOLOR_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the value of a cell once formulas have been calculated.

Data type

Variant

Syntax

value **=** *range***.CellValue**

Legal values

The value of the CellValue property is the value that appears in the specified cell.

```
' Example: CellValue property
' Return the value of a cell.
Dim myvalue As Variant
[A:B9].Contents = "10*5"
myvalue = [A:B9].CellValue
Msgbox "The value of B9 is: " + myvalue
' The result will be a messagebox that says, "The value of B9 is: 50".
```

1-2-3: CenterLeftToRight property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether data and graphic objects will be centered horizontally when printed.

Data type

Variant (Boolean)

Syntax

printsettings.**CenterLeftToRight =** *value*

value **=** *printsettings*.**CenterLeftToRight**

Legal values

Value	Description
TRUE	Center printed data horizontally.
FALSE	(Default) Do not center printed data.

1-2-3: CenterTopToBottom property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether data will be centered vertically when printed.

Data type

Variant (Boolean)

Syntax

printsettings.**CenterTopToBottom =** *value*

value **=** *printsettings*.**CenterTopToBottom**

Legal values

Value	Description
TRUE	Center printed data vertically.
FALSE	(Default) Do not center printed data.

1-2-3: CenturyLongFormat property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_CENTURYLONGFORMAT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Determines whether the year number in date formats have a long format. For example, 1997 is a long format, while 97 is a short format.

Data type

Variant (Boolean)

Syntax

value **=** *application*.**CenturyLongFormat**

Legal values

Value	Description
TRUE	The year number has a long format.
FALSE	(Default) The year number has a short format.

```
' Example: CenturyLongFormat property
' Determine what format the operating system is using to display the year.
Dim dispdate as Variant
dispdate = CurrentApplication.CenturyLongFormat
Print dispdate
'This prints True if the operating system is set to show the year in four digits,
'or False if the operating system is set to show the year in two digits.
```

1-2-3: Changed property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_CHANGED_PROPERTY_EXSCRIPT',1)} See example

Determines whether the file has been changed since it was last saved.

Data type

Variant (Boolean)

Syntax

document.**Changed =** *value*

value **=** *document*.**Changed**

Legal values

Value	Description
TRUE	The file has changed.
FALSE	The file has not changed.

```
' Example: Changed property
' Determine whether a file has changed or not.
Dim docchanged As Variant
docchanged = CurrentDocument.Changed
' Show the mesage True if the file has changed and
' False if the file has not changed.
MessageBox Str(docchanged)
```

1-2-3: ChartsPicturesAndDrawPrint property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether graphic objects should be printed with sheet data.

Data type

Variant (Boolean)

Syntax

printsettings.**ChartPicturesAndDrawPrint =** *value*

value **=** *printsettings*.**ChartPicturesAndDrawPrint**

Legal values

Value	Description
TRUE	(Default) Print graphic objects with sheet data.
FALSE	Do not print graphic objects with sheet data.

1-2-3: Charts property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

(Read-only) Returns a ChartsCollection object.

Data type

Charts

Syntax

Set *charts* **=** *document***.Charts**

Legal values

The value for the Charts property is the collection of charts in the specified file.

1-2-3: ClassicMenuActivationKey property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the key(s) you use to display the 1-2-3 Classic menu.

Data type

Variant (ClassicMenuKeyChoices enumeration)

Syntax

application.**ClassicMenuActivationKey =** *value*

value **=** *application*.**ClassicMenuActivationKey**

Legal values

Value	Description
$SlashKey	(Default) Slash key displays the Classic menu.
$LessThanKey	Angle bracket displays the Classic menu.
$SlashAndLessThanKey	Either the slash key or the angle bracket displays the Classic menu.

1-2-3: ClassicMenuEnabled property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether the user can turn on the display of the 1-2-3 Classic menu.

Data type

Variant (Boolean)

Syntax

application.**ClassicMenuEnabled =** *value*

value **=** *application*.**ClassicMenuEnabled**

Legal values

Value	Description
TRUE	(Default) The user can turn on the display of the 1-2-3 Classic menu.
FALSE	The user can turn off the display of the 1-2-3 Classic menu.

1-2-3: ClassName property

{button ,AL(`H_123_CLASSINFO_CLASS',0)} See list of classes

(Read-only) Returns the name of the data type of the specified class.

Data type

String

Syntax

value **=** *classinfo***.ClassName**

Legal values

The value of the ClassName property is the name of any valid 1-2-3 class.

1-2-3: ClassVersionId property

{button ,AL(`H_123_CLASSINFO_CLASS',0)} See list of classes

(Read-only) Returns the version of the specified object.

Data type

Long

Syntax

value **=** *classinfo***.ClassVersionId**

Legal values

The default value of the ClassVersionId property is the version of the specified object.

1-2-3: Class property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_QUERYTABLE_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_RANGESELECTOR_CLASS;',0)} See list of classes

{button ,AL(`H_123_CLASS_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the ClassInfo subobject for the specified object.

Data type

ClassInfo

Syntax

Set *classinfo* **=** *object*.**Class**

Legal values

The value of the Class property is the ClassInfo subobject for the specified object.

```
' Example: Class property
' To return the class type of the selection.
MessageBox "The class type of the selection is " + Selection.Class.Classname
```

1-2-3: Collate property

{button ,AL(`H_123_PRINTSETTINGS_CLASS;',0)} See list of classes

Determines whether the print selection is collated or not. This property applies to printers with the collate function only.

Data type

Variant (Boolean)

Syntax

printsettings.**Collate =** *value*

value **=** *printsettings*.**Collate**

Legal values

Value	Description
TRUE	Collate the print selection.
FALSE	(Default) Do not collate the print selection.

1-2-3: ColorBins property

{button ,AL(`H_123_DRAWCOLLECTION_CLASS;H_123_MAP_CLASS;H_123_GROUP_CLASS',0)} See list of classes

(Read-only) Returns the collection of color bins for a map.

Data type

MapBins

Syntax

Set *mapbins* **=** *object*.**ColorBins**

Legal values

The value for the ColorBins property is a MapBins object.

1-2-3: ColorIndex property

{button ,AL(`H_123_COLOR_CLASS',0)} See list of classes

{button ,AL(`H_123_COLORINDEX_PROPERTY_EXSCRIPT;H_123_NEGATIVESINCOLOR_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the palette index of the color.

Data type

Long

Syntax

object.color.**ColorIndex** = *value*

value = *object.color*.**ColorIndex**

Legal values

Any Long from 0 - 239. For a list of color names and color index values, see the Color palette.

```
' Example: BackColor, Background, and ColorIndex properties
' To set the background to every available color.
Dim palindex As Long
'Set the background to every available color.
For palindex = 239 To 0 Step -1
[A:A1].Background.BackColor.ColorIndex = palindex
Next palindex
```

1-2-3: ColorName property

{button ,AL(`H_123_COLOR_CLASS;',0)} See list of classes

{button ,AL(`H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;H_123_SHOWDRAWLAYER_PROPERTY_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT;H_123_EDGECOLOR_PROPERTY_EXSCRIPT;H_123_COLORNAME_PROPERTY_EXSCRIPT;H_123_PATTERN_PROPERTY_EXSCRIPT;H_123_FONTCOLOR_PROPERTY_EXSCRIPT;H_123_GRIDBORDER_PROPERTY_EXSCRIPT;H_123_HORIZONTALBORDER_PROPERTY_EXSCRIPT;H_123_INNERBORDER_PROPERTY_EXSCRIPT;',1)} See example

Set or returns the name of the color of the Color object.

Data type

String

Syntax

object.*color*.**ColorName =** *value*

value **=** *object*.*color*.**ColorName**

Legal values

The value for the ColorName property is the name of the color of the Color object.

For a list of all the color names, see the Color palette.

Usage

This string is identical to the Name property of the color.

```
' Example: ColorName, Font, and FontColor properties
' Select a range and change font color.
Sub FontColor
   [A:B5..A:H25].Select
   [A:A5..A:H25].Font.FontColor.ColorName = "Wedgewood"
End Sub
```

1-2-3: Colors property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_BACKCOLOR_PROPERTY_EXSCRIPT;H_123_COLORS_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the collection of 240 colors.

Data type

Colors

Syntax

Set *colors* **=** *application***.Colors**

Legal values

The values of the Colors property are Color objects.

Usage

You can access the Color objects in the Colors property either by the name of the color (a string index) or by numbers 0 - 239 (an ordinal index). See Color palette for a list of color names and the color index.

```
' Example: Background, BackColor, and Colors properties;
' CreateRangeName method
' Create a range name, then apply a specific color to that range.
' First, create the range.
CurrentDocument.CreateRangeName "North Sales", [A:B2..A:B10]
' Then, apply the color.
Set [North Sales].Background.BackColor = CurrentApplication.Colors("light yellow")
```

1-2-3: ColorVisible property

{button ,AL(`H_123_LEGEND_CLASS',0)} See list of classes

Determines whether the color legend for a map is displayed or not.

Data type

Variant (Boolean)

Syntax

legend.**ColorVisible =** *value*

value **=** *legend*.**ColorVisible**

Legal values

Value	Description
TRUE	(Default) Display the map's color legend.
FALSE	Do not display the map's color legend.

1-2-3: Color palette

The Colors collection holds a collection of 240 color constants, representing all the colors in the palette. You can access the Colors collection by an ordinal index (0 - 239) or by a string index (the name of the color in English in all releases of 1-2-3). The following table shows the index number and name for all the colors in the palette.

See the Color property for information on specifying colors.

Color Index	Color Name
0	white
1	vanilla
2	parchment
3	ivory
4	pale green
5	sea mist
6	ice blue
7	powder blue
8	arctic blue
9	lilac mist
10	purple wash
11	violet frost
12	seashell
13	rose pearl
14	pale cherry
15	white
16	blush
17	sand
18	light yellow
19	honeydew
20	celery
21	pale aqua
22	pale blue
23	crystal blue
24	lt cornflower
25	pale lavender
26	grape fizz
27	pale plum
28	pale pink
29	pale rose
30	rose quartz
31	5% gray
32	red sand
33	buff
34	lemon
35	pale lemon lime
36	mint green
37	pastel green
38	pastel blue

39	sapphire
40	cornflower
41	light lavender
42	pale purple
43	light orchid
44	pink orchid
45	apple blossom
46	pink coral
47	10% gray
48	light salmon
49	light peach
50	yellow
51	avocado
52	leaf green
53	light aqua
54	lt turquoise
55	light cerulean
56	azure
57	lavender
58	light purple
59	dusty violet
60	pink
61	pastel pink
62	pastel red
63	15% gray
64	salmon
65	peach
66	mustard
67	lemon lime
68	neon green
69	aqua
70	turquoise
71	cerulean
72	wedgewood
73	heather
74	purple haze
75	orchid
76	flamingo
77	cherry pink
78	red coral
79	20% gray
80	dark salmon
81	dark peach
82	gold
83	yellow green

84	light green
85	caribbean
86	dk pastel blue
87	dark cerulean
88	manganese blue
89	lilac
90	purple
91	lt red violet
92	light magenta
93	rose
94	carnation pink
95	25% gray
96	watermelon
97	tangerine
98	orange
99	chartreuse
100	green
101	teal
102	dark turquoise
103	lt slate blue
104	medium blue
105	dark lilac
106	royal purple
107	fuchsia
108	confetti pink
109	pale burgundy
110	strawberry
111	30% gray
112	rouge
113	burnt orange
114	dark orange
115	light olive
116	kelly green
117	sea green
118	aztec blue
119	dusty blue
120	blueberry
121	violet
122	deep purple
123	red violet
124	hot pink
125	dark rose
126	poppy red
127	35% gray
128	crimson

129	red
130	light brown
131	olive
132	dark green
133	dark teal
134	spruce
135	slate blue
136	navy blue
137	blue violet
138	amethyst
139	dk red violet
140	magenta
141	light burgundy
142	cherry red
143	40% gray
144	dark crimson
145	dark red
146	hazelnut
147	dark olive
148	emerald
149	malachite
150	dark spruce
151	steel blue
152	blue
153	iris
154	grape
155	plum
156	dark magenta
157	burgundy
158	cranberry
159	50% gray
160	mahogany
161	brick
162	dark brown
163	deep olive
164	dark emerald
165	evergreen
166	baltic blue
167	blue denim
168	cobalt blue
169	dark iris
170	midnight
171	dark plum
172	plum red
173	dark burgundy

174	scarlet
175	60% gray
176	chestnut
177	terra cotta
178	umber
179	amazon
180	peacock green
181	pine
182	metallic blue
183	dk slate blue
184	royal blue
185	lapis
186	dark grape
187	aubergine
188	dark plum red
189	raspberry
190	deep scarlet
191	70% gray
192	burnt sienna
193	milk chocolate
194	burnt umber
195	deep avocado
196	deep forest
197	dark pine
198	dk metalic blue
199	air force blue
200	ultramarine
201	prussian blue
202	raisin
203	eggplant
204	boisenberry
205	bordeaux
206	ruby
207	75% gray
208	red gray
209	tan
210	khaki
211	putty
212	bamboo green
213	green gray
214	baltic gray
215	blue gray
216	rain cloud
217	lilac gray
218	lt purple gray

219	light mauve
220	lt plum gray
221	lt burgundy gray
222	rose gray
223	80% gray
224	dark red gray
225	dark tan
226	safari
227	olive gray
228	jade
229	dk green gray
230	spruce gray
231	dk blue gray
232	atlantic gray
233	dk lilac gray
234	purple gray
235	mauve
236	plum gray
237	burgundy gray
238	dark rose gray
239	black

1-2-3: Color property

{button ,AL(`H_123_BACKGROUND_CLASS;H_123_MAPBIN_CLASS;H_123_RANGEBORDER_CLASS;',0)} See list of classes

{button ,AL(`H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;H_123_SHOWDRAWLAYER_PROPERTY_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT;H_123_COLOR_PROPERTY_EXSCRIPT;H_123_GRIDBORDER_PROPERTY_EXSCRIPT;H_123_HORIZONTALBORDER_PROPERTY_EXSCRIPT;H_123_INNERBORDER_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the color object of the specified object.

Data type

Color

Syntax

Set *object*.**Color =** *color*

Set *color* **=** *object*.**Color**

Legal values

You can use the Color property in conjunction with the ColorName or ColorIndex property. See the Color palette for a list of the color names and index values. You can also specify any color object from the Colors collection, as shown here:

```
Dim y As Color
Set y = CurrentApplication.Colors("red")
Set [c:b12].BottomBorder.Color = y
```

Usage

The value of the Color property for a graphic object is the current pattern color. The value of the Color property for a bin object is the current color for Color type bins.

```
' Example: Color and Colors properties
' Set the Color value.
Dim mycolor As Variant
Set mycolor = CurrentApplication.Colors("magenta")
```

1-2-3: ColumnFolding property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_COLUMNFOLDING_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the location of the parent or child items in the outline of a column. The parent items can be located either before or after the children items in an outline.

Data type

Variant (FoldDir enumeration)

Syntax

sheet.**ColumnFolding =** *value*

value **=** *sheet*.**ColumnFolding**

Legal values

Value	Description
$ParentBefore	(Default) Display the parent item before the child item.
$ParentAfter	Display the parent item after the child item.

```
' Example: ColumnFolding and ColumnOutlineVisible properties
' Display the column outline, with the parent items displayed to the right.
' Demote a column to show the location of the parent.
Sub ShowColumn
       [A].ColumnOutlineVisible = True
       [A].ColumnFolding = $ParentAfter
       [A:F8].Select
       Selection.DemoteColumn
End Sub
```

1-2-3: ColumnOutlineVisible property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_COLUMNOUTLINEVISIBLE_PROPERTY_EXSCRIPT',1)} See example

Determines whether the outline frame above the column letters is displayed or not.

Data type

Variant (Boolean)

Syntax

sheet.**ColumnOutlineVisible =** *value*

value **=** *sheet*.**ColumnOutlineVisible**

Legal values

Value	Description
TRUE	Display the outline frame above the column letters.
FALSE	(Default) Do not display the outline frame above the column letters.

```
' Example: ColumnOutlineVisible property
' Display the outline frame above the column letters.
[A].ColumnOutlineVisible = True
```

1-2-3: ColumnTitleRange property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns a print title range, identified as a column only. This print title range is printed to the left of the print range on every printed page.

Data type

Variant

Syntax

printsettings.**ColumnTitleRange =** *value*

value **=** *printsettings*.**ColumnTitleRange**

Legal values

The value for the ColumnTitleRange property is the column you set or the column you specify using File - Preview & Page Setup (Headers & Footers).

1-2-3: ColumnWidth property

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_COLUMNWIDTH_PROPERTY_EXSCRIPT;H_123_PATTERN_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the width of all columns in the specified range.

Data type

Long

Syntax

range.**ColumnWidth =** *value*

value **=** *range*.**ColumnWidth**

Legal values

The value of the ColumnWidth property is any long from 0 - 240.

Usage

Setting the column width to 0 hides the column.

```
' Example: ColumnWidth property
' Select a range and adjust its column width.
Sub AdjustWidth
   [A:B5..A:H25].select
   [A:B5..A:H25].ColumnWidth = 23
End Sub
```

1-2-3: ConfirmDragAndDrop property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether confirmation of drag and drop is enabled.

Data type

Variant (Boolean)

Syntax

application.**ConfirmDragAndDrop =** *value*

value = *application*.**ConfirmDragAndDrop**

Legal values

Value	Description
TRUE	(Default) Enable confirmation of drag and drop.
FALSE	Do not enable confirmation of drag and drop.

1-2-3: Contents property

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_BACKCOLOR_PROPERTY_EXSCRIPT;H_123_SHOWVERSIONBORDERS_PROPERTY_EXSCRIPT;H_123_DATAPROTECTED_PROPERTY_EXSCRIPT;H_123_CONTENTS_PROPERTY_EXSCRIPT;H_123_GREEN_PROPERTY_EXSCRIPT;H_123_KEYWORDS_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the contents of a cell as it appears in the edit window.

Data type

String

Syntax

range.**Contents =** *value*

value **=** *range*.**Contents**

Legal values

When referring to a multi-cell range, .Contents refers to the cell in the upper left corner of the range only.

```
' Example: Contents property
' Set the contents of a cell.
[A:B2].Contents = "Field 1"
```

1-2-3: CoordinateRange property

{button ,AL(`H_123_DRAWCOLLECTION_CLASS;H_123_MAP_CLASS;H_123_GROUP_CLASS;',0)} See list of classes

Sets or returns the range that contains the pin character data for a map.

Data type

Variant

Syntax

object.**CoordinateRange =** *value*

value **=** *object*.**CoordinateRange**

Legal values

The value for the CoordinateRange property is the range you set or the range you specified using Map - Ranges.

1-2-3: CoordinateString property

{button ,AL(`H_123_RANGE_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS;H_123_ISRANGENAMED_PROPERTY_EXSCRIPT',0)} See list of classes

(Read-only) Returns a coordinate string for the specified range.

Data type

String

Syntax

value **=** *range*.**CoordinateString**

Legal values

The value of the CoordinateString property is a string containing a valid range, for example "A:A1..B:B12". The CoordinateString property never returns a range name.

1-2-3: Copies property

{button ,AL(`H_123_PRINTSETTINGS_CLASS;',0)} See list of classes

Sets or returns the number of copies to be printed.

Data type

Long

Syntax

printsettings.**Copies =** *value*

value **=** *printsettings*.**Copies**

Legal values

The value for the Copies property is the number of copies to be printed you set or the number of copies to be printed that you specified using File - Print.

1-2-3: CountryCode property

{button ,AL(`H_123_APPLICATION_CLASS;',0)} See list of classes

(Read-only) Returns the country code, using the international telephone code. The country code used for Canada is 2.

Data type

Long

Syntax

value **=** *application*.**CountryCode**

Legal values

The value for the CountryCode property can be any international telephone code.

1-2-3: Count property

{button ,AL(`;H_123_BASECOLLECTION_CLASS;H_123_CHARTS_CLASS;H_123_COLORS_CLASS;H_123_DATALINKS_CLASS;H_123_DOCUMENTS_CLASS;H_123_DOCWINDOWS_CLASS;H_123_DRAWOBJECTS_CLASS;H_123_MAPBINS_CLASS;H_123_MAPS_CLASS;H_123_MAPTEXTENTRIES_CLASS;H_123_OLEOBJECTS_CLASS;H_123_PRINTSETTINGSCOLLECTION_CLASS;H_123_QUERYTABLES_CLASS;H_123_RANGES_CLASS;H_123_SHEETS_CLASS;H_123_STRINGS_CLASS;H_123_VERSIONGROUPS_CLASS;H_123_VERSIONS_CLASS;H_123_WINDOWS_CLASS;H_123_DRAWCOLLECTION_CLASS',0)} See list of classes

{button ,AL(`H_123_SHEETS_PROPERTY_EXSCRIPT;H_123_COUNT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the number of items in the specified collection.

Data type

Long

Syntax

value **=** *object***.Count**

Legal values

The value for the Count property is the number of items in the specified collection.

```
' Example: Count, Name, and NamedRanges properties
' Count the number of named ranges in a file.
Msgbox "There are "+Cstr(CurrentDocument.NamedRanges.Count)+" named ranges in the file
"+CurrentDocument.Name
```

1-2-3: CreationDate property

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;',0)} See list of classes

{button ,AL(`H_123_CREATIONDATE_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the date the specified file was created.

Data type

DateTime

Syntax

Set *datetime* **=** *object*.**CreationDate**

Legal values

The value for the CreationDate property is the date the file or file version was created.

```
' Example: CreationDate property
' To return the date the document was created.
Msgbox "This document was created on " + _
        CurrentDocument.CreationDate.LocalTime
```

1-2-3: Criteria property

{button ,AL(`H_123_QUERY_CLASS;',0)} See list of classes

Sets or returns a formula that represents the criteria that determine which results appear in a query output location.

Data type

String

Syntax

dataquery.**Criteria =** *value*

value **=** *dataquery*.**Criteria**

Legal values

The default value of the Criteria property is " ", which returns all records.

For the requirements of a valid criteria formula, search on "Criteria" in the 1-2-3 Help Index.

1-2-3: CurrentDirectory property

{button ,AL(`H_123_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_123_CURRENTDIRECTORY_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the directory used by 1-2-3 to save workbook files for the current session.

Data type

String

Syntax

application.**CurrentDirectory** = *value*

value = *application*.**CurrentDirectory**

Legal values

The value of the CurrentDirectory property is a string containing a valid directory name, for example "C:\LOTUS\WORK\123".

Usage

The value for the CurrentDirectory property is not saved between 1-2-3 sessions. It is reset to the value of DefaultPath.

```
' Example: CurrentDirectory property
' Set the directory that 1-2-3 uses to save work files.
CurrentApplication.CurrentDirectory = "C:\LOTUS\WORK\123"
```

1-2-3: CurrentMenuBar property

{button ,AL(`H_123_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_123_ADDMENU_METHOD_EXSCRIPT;H_123_DISABLEITEM_METHOD_EXSCRIPT;H_123_CURRENTMENUBAR_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the current menu bar.

Data type

MenuBar

Syntax

Set *application*.**CurrentMenuBar** = *menubar*

Set *menubar* = *application*.**CurrentMenuBar**

Legal values

The value for the CurrentMenuBar property is any valid MenuBar object.

```
' Example: CurrentMenuBar property
' Get a reference to the application's top-level menu bar.
Dim mymenubar As Variant
Set mymenubar = CurrentApplication.CurrentMenuBar
```

1-2-3: CurrentPrintSettings property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_CURRENTPRINTSETTINGS_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the print settings for a document.

Data type

PrintSettings

Syntax

Set *document*.**CurrentPrintSettings =** *printsettings*

Set *printsettings* **=** *document*.**CurrentPrintSettings**

Legal values

The value of the CurrentPrintSettings property is the current PrintSettings object. See the File - Preview & Page Settings InfoBox for a description of the print features you can set.

Usage

The current print settings are persistent and are saved with the workbook file.

```
' Example: CurrentPrintSettings property; NewNamedPrintSettings method
' Create a PrintSettings object, set a print property, and set the
' new PrintSettings object to be the current print setting.
Dim myprintset As PrintSettings
Set myprintset = CurrentDocument.NewNamedPrintSettings("newps")
myprintset.Orientation = $Landscape
Set CurrentDocument.CurrentPrintSettings = myprintset
```

1-2-3: CurrentSheet property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_SHEETNUMBER_PROPERTY_EXSCRIPT;H_123_CURRENTSHEET_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns the current sheet.

Data type

Sheet

Syntax

Set *sheet* **=** *document*.**CurrentSheet**

Legal values

The value for the CurrentSheet property is the current sheet.

```
' Example: CurrentSheet and SheetNumber properties
Sub NewSheet
   ' Declare variables
   Dim current As Sheet
   Dim sheetnum As Long

   ' Get the current sheet, and add another sheet after it.
   Set current = CurrentDocument.CurrentSheet
   CurrentDocument.NewSheet $After,1,True
   sheetnum = Current.SheetNumber

   ' Set the current sheet to be the sheet you just added, and select it.
   Set current = CurrentDocument.CurrentSheet
   Current.Select

End Sub
```

1-2-3: CurrentVersion property

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_CURRENTVERSION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the current version of the specified range.

Data type

Range

Syntax

Set *range* **=** *range*.**CurrentVersion**

Legal values

The value for the CurrentVersion property is the current version of the specified range.

```
' Example: CurrentVersion, Description, and Name properties
' Creates a few versions, then do some work on the current version.
Sub Click(Source As ButtonControl)
   Dim v As Version
'  Create a named range.
   [A:A1..A:B5].Name = "RANGE 1"
   [RANGE 1].NewVersion "Best Case"
   [RANGE 1].NewVersion "Worst Case"
   Set v = [RANGE 1].CurrentVersion
   v.Description = "This is the worst possible case we will encounter!"
   v.Share = $Protected
End Sub
```

1-2-3: DataLinks property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_DATALINKS_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the collection of DataLink objects in the specified file.

Data type

DataLinks

Syntax

Set *datalinks* **=** *document*.**DataLinks**

Legal values

The default value for the DataLinks property is the collection of DataLink objects in the specified file.

```
' Example: Count, DataLinks, and Name properties; Item method
Sub MyLinks

   ' Create a string to hold information about the links.
   Dim mystring As String
   mystring = "Names of current links: "

   ' Show the number of links in the file.
   Messagebox "Number of data links in current file: " +
   Str(CurrentDocument.DataLinks.Count)

   If CurrentDocument.DataLinks.Count = 0 Then
       mystring = "There are no links in the current file"
   Else
       ' Build up mystring, putting the name of each link in the string.
       For i = 0 To CurrentDocument.DataLinks.Count - 1
         mystring = mystring + CurrentDocument.DataLinks.Item(i).Name + " "
       Next
   End If

   ' Display mystring, which contains either a list of all links in the string or the
   ' text "There are no links in the current file".
   Messagebox mystring

End Sub
```

1-2-3: DataProtected property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_DATAPROTECTED_PROPERTY_EXSCRIPT;',1)} See example

Determines whether cell contents in a file are protected or not.

Data type

Variant (Boolean)

Syntax

document.**DataProtected =** *value*

value **=** *document*.**DataProtected**

Legal values

Value	Description
TRUE	Protect cell contents from changes.
FALSE	(Default) Do not protect cell contents from changes.

```
' Example: Contents, DataProtected, and IsProtected properties
Sub SetProtection
   CurrentApplication.NewDocument

   ' By default, all cells are locked. Verify this by
   ' returning the value of the IsProtected property for a cell.
   Messagebox "Value of IsProtected: " + [A:A1].IsProtected

   ' You can still write to the cell, though.
   [A:A3].Contents = "Hello"

   ' Now set DataProtected to true.
   CurrentDocument.DataProtected = True

   ' You can no longer write to the cell; trying to do so would
   ' give an error message.
   ' Instead, you have to make the cells you want to write to unprotected.
   ' 1-2-3 writes the text in a different color to show
   ' that it's not protected.
   [A:A4].IsProtected = False
   [A:A4].Contents = "Good-bye"

   ' Turn off data protection again.
   CurrentDocument.DataProtected = False

End Sub
```

1-2-3: DateOrder property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DATEORDER_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the order in which dates are written.

Data type

Variant (DateOrder enumeration)

Syntax

Set *value* **=** *application*.**DateOrder**

Legal values

Value	Description
$MDY	Write dates as month/day/year (for example, 05/28/97).
$DMY	Write dates as day/month/year (for example, 28/05/97).
$YMD	Write dates as year/month/day (for example, 97/05/28).

```
' Example: DateOrder, DefaultDecimals, and DefaultNegCurrencyFormat properties
Sub CountrySettings
   Dim order, showorder, showformat As String

   ' Determine how the date order is set in the regional (country) settings.
   order = CurrentApplication.DateOrder

   ' Translate the date order code to a string that makes sense to the user.
   Select Case order
     Case $MDY : showorder = "Month/Day/Year"
     Case $DMY : showorder = "Day/Month/Year"
     Case $YMD : showorder = "Year/Month/Day"
     Case Else : showorder = "not set"
   End Select

   Msgbox "Date order is: " + showorder

   ' Determine how negative currency is shown.
   If CurrentApplication.DefaultNegCurrencyFormat = $Parens Then
     showformat = "shown with parentheses"
   Else
     showformat = "shown with a minus sign"
   End If

   ' Display a message box saying how the currency is shown.
   Msgbox "Negative currency is " + showformat

   ' Determine the number of decimal places used when displaying the number format.
   Msgbox "Number of decimal places is " + Str(CurrentApplication.DefaultDecimals)


End Sub
```

1-2-3: DateSeparator property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DATESEPARATOR_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the character used to separate date fields when a date format is used. This character is set in the Windows Control Panel.

Data type

String

Syntax

value **=** *application*.**DateSeparator**

Legal values

The value of the DateSeparator property is set in the Windows Control Panel.

```
' Example: DateSeparator and DecimalSeparator properties
Sub CheckDefaults
   Dim msgstring As String

   ' Create a two-line message (using the vertical bar to
   ' include a line feed) with the date and decimal separator.
   msgstring = "Date separator: " + CurrentApplication.DateSeparator + |
|+ "Decimal separator: " + CurrentApplication.DecimalSeparator

   Msgbox msgstring
End Sub
```

1-2-3: DayNames property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DAYNAMES_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns a collection of strings that represent the unabbreviated days of the week.

Data type

Strings

Syntax

strings **=** *application*.**DayNames**

Legal values

The value for the DayNames property is the collection of strings.

```
' Example: Count, DayNames, Name, and Sheet properties; Item method
' This example creates a new document, adds 6 sheet tabs, and names
' each tab with a day of the week.
Sub NewTabs
   Dim mydoc As Document
   Dim mysheet As Sheet
   Dim x As Long

   ' Create a new workbook.
   Set mydoc = CurrentApplication.NewDocument

   ' Add 6 new tabs to the document; don't make the last sheet tab the selected tab.
   mydoc.NewSheet $After,6,False

   ' Make each sheet tab have a day of the week.
   For x = 0 To (mydoc.Sheets.Count - 1)
     mydoc.Sheets.Item(x).Name = CurrentApplication.DayNames.Item(x)
   Next x

End Sub
```

1-2-3: DecimalSeparator property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DATESEPARATOR_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the character used to separate decimals. This character is set in the Windows Control Panel.

Data type

String

Syntax

value **=** *application*.**DecimalSeparator**

Legal values

The legal value for the DecimalSeparator property can be any single character set through the WIndows control panel.

1-2-3: DefaultAddinPath property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFAULTPATH_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the default path used to load an add-in if no path is specified.

Data type

String

Syntax

application.**DefaultAddinPath =** *value*

value **=** *application*.**DefaultAddinPath**

Legal values

The value for the DefaultAddinPath property is the path the user specified using File - User Setup - 1-2-3 Preferences (File Locations). The default value for add-ins is \Lotus\123\Addins.

1-2-3: DefaultColumnWidth property

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_DISPLAYZEROAS_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the default column width for the specified sheet.

Data type

Long

Syntax

value **=** *sheet*.**DefaultColumnWidth**

Legal values

The value for the DefaultColumnWidth property is any long from 1 - 240.

1-2-3: DefaultDecimals property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DATEORDER_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the number of decimal places used when displaying the number format.

Data type

Long

Syntax

value **=** *application*.**DefaultDecimals**

Legal values

The value for the DefaultDecimals property is set in the Windows Control Panel.

1-2-3: DefaultFileExtension property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFAULTPATH_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the default extension to apply when saving or listing files.

Data type

String

Syntax

application.**DefaultFileExtension =** *value*

value **=** *application*.**DefaultFileExtension**

Legal values

The value for the DefaultFileExtension property is .123, or whatever value you set using the 1-2-3 Classic menu (/Worksheet Global Default Ext Save).

1-2-3: DefaultFontName property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFAULTFONT_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the default font you want 1-2-3 to use for new files.

Data type

String

Syntax

application.**DefaultFontName =** *value*

value **=** *application*.**DefaultFontName**

Legal values

The value of the DefaultFontName property is the font you set or the font you specified using File - User Setup - 1-2-3 Preferences (New Workbook Defaults). For a US installation, the default value is Arial.

1-2-3: DefaultFontSize property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFAULTFONT_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the default font point size you want 1-2-3 to use for new files.

Data type

Long

Syntax

application.**DefaultFontSize =** *value*

value **=** *application*.**DefaultFontSize**

Legal values

The value of the DefaultFontSize property is the point size you set or the point size you specified using File - User Setup - 1-2-3 Preferences (New Workbook Defaults). For a US installation, the default value is 12.

```
' Example: ColorName, DefaultFontName, DefaultFontSize, and DefaultTextColor
properties
Sub MakeNew
   Dim answer As Long
   Dim myapp As Application
   Set myapp = CurrentApplication

   ' 4 in the following statement means a YesNo box;
   ' 32 means display a question mark.
   answer = Messagebox("Use custom font and color settings?",4 + 32)

   ' An answer of 6 means the user clicked "Yes".
   If answer = 6 Then
       myapp.DefaultFontName = "Arial"
       myapp.DefaultTextColor.ColorName = "deep purple"
       myapp.DefaultBackColor.ColorName = "ivory"
       myapp.DefaultFontSize = 12
       myapp.UseOSDefaultColors = False
   Else
       Messagebox("Using current workbook defaults for new workbook...")
   End If

   ' Create a new workbook.
   CurrentApplication.NewDocument

End Sub
```

1-2-3: DefaultNegCurrencyFormat property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DATEORDER_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the default negative currency format used when displaying numbers in a currency format.

Data type

Variant (NegCurVal enumeration)

Syntax

value **=** *application*.**DefaultNegCurrencyFormat**

Legal values

The value of the DefaultNegCurrencyFormat property is set in the Windows Control Panel.

Value	Description
$Parens	Negative values are in parentheses.
$Minus	(Default) Negative values are displayed with a minus sign.

1-2-3: DefaultPath property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFAULTPATH_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the default folder for Workbook files if no other folder is specified.

Data type

String

Syntax

application.**DefaultPath =** *value*

value **=** *application*.**DefaultPath**

Legal values

The default value of the DefaultPath property is "\Lotus\Work\123\".

```
' Example: DefaultAddinPath, DefaultFileExtension, and DefaultPath properties
Sub ChangeDefaults
   Dim extension As String, path As String, addinpath As String

   ' Change the extension to be what the user specifies.
   ' If the user presses Cancel, don't change the extension.
   extension = Inputbox("Specify the file extension to use: "_
   ,,CurrentApplication.DefaultFileExtension)
   If extension <> "" Then
       CurrentApplication.DefaultFileExtension = extension
   End If

   ' Change the default path to be what the user specifies.
   ' If the user presses Cancel, don't change the path.
   path = Inputbox("Specify the path to look for workbooks: "_
   ,,CurrentApplication.DefaultPath)
   If path <> "" Then
       CurrentApplication.DefaultPath = path
   End If

   ' Change the default add-in path to be what the user specifies.
   ' If the user presses Cancel, don't change the add-in path.
   addinpath = Inputbox("Specify the path to look for add-ins: "_
   ,,CurrentApplication.DefaultAddinPath)
   If addinpath <> "" Then
       CurrentApplication.DefaultAddinPath = addinpath
   End If

End Sub
```

1-2-3: DefaultRowHeight property

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_DISPLAYZEROAS_PROPERTY_EXSCRIPT;H_123_ROWHEIGHTUSEFONTSIZE_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the default row height for the specified sheet.

Data type

Long

Syntax

sheet.**DefaultRowHeight =** *value*

value **=** *sheet*.**DefaultRowHeight**

Legal values

The value of the DefaultRowHeight property is the row height you set or the row height you specified using Sheet - Sheet Properties (Basics). Legal values are from 1 - 255.

1-2-3: DefaultTextColor property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_DEFAULTFONT_PROPERTY_EXSCRIPT;',1)} See example

Set or returns the default text color for new workbooks.

Data type

Color

Syntax

application.**DefaultTextColor** = *color*

color = *application*.**DefaultTextColor**

Legal values

See Color palette for a list of the color names and color index.

```
' Example: CalcMode and Description properties; Calc method
' Create a new document and give a description of it.
Dim doc1 As Document
Set doc1 = CurrentApplication.NewDocument("Doc_1")
CurrentDocument.Description = "first in a series with calculations"
' Set up a summed range.
[A:B2].Contents = "1"

[A:B3].Contents = "2"

[A:B4].Contents = "3"
MessageBox "Add the column"
[A:B5].Contents = "@Sum(B2..B4)"
' Set Calc mode to manual, to control when recalculation occurs.
CurrentDocument.CalcMode = $Manual
' B5 now shows 6.  Now change B2.
MessageBox "Change 1 to 2"
[A:B2].Contents = "2"
' To see the effect of this, you must recalculate, because
' you set .CalcMode = $Manual.
MessageBox "Recalculate"
CurrentApplication.Calc
' B5 now shows 7.
```

1-2-3: Description property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_BUTTON_CLASS;H_123_ARC_CLASS;H_123_CHART_CLASS;H_123_DATALINK_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_DESCRIPTION_PROPERTY_EXSCRIPT;H_123_EDITINGTIME_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns comments or a description for the specified object.

Data type

String

Syntax

object.**Description =** *value*

value **=** *object*.**Description**

Legal values

The value of the Description property is a string containing up to 1024 characters.

Usage

The value of the Description property is set when the user enters text in a "Comments" or "Description" field. This property has no effect for objects other than Document, Version, or VersionGroup.

1-2-3: DesignerFrameStyle property

{button ,AL(`;H_123_APPROACHCONNECTION_CLASS;H_123_CHART_CLASS;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

Sets or returns the style of a designer frame.

Data type

Variant (DesignerFrameStyle enumeration)

Syntax

object.**DesignerFrameStyle =** *value*

value **=** *object*.**DesignerFrameStyle**

Legal values

The value for the DesignerFrameStyle property can also be set in the Range - Range Properties infobox.

Value	Description
$DesignerFrame1	
$DesignerFrame2	
$DesignerFrame3	
$DesignerFrame4	
$DesignerFrame5	
$DesignerFrame6	
$DesignerFrame7	
$DesignerFrame8	
$DesignerFrame9	
$DesignerFrame10	
$DesignerFrame11	
$DesignerFrame12	
$DesignerFrame13	
$DesignerFrame14	
$DesignerFrame15	
$DesignerFrame16	

1-2-3: Display4DigitYear property

{button ,AL(`H_123_APPLICATION_CLASS;',0)} See list of classes

Sets or returns whether to display the year as a 4-digit number.

Data type

Variant (Boolean)

Syntax

application.**Display4DigitYear =** *value*

value **=** *application*.**Display4DigitYear**

Legal values

Value	Description
TRUE	Display the year as a 4-digit number.
FALSE	Display the year as a 2-digit number.

1-2-3: DisplayZeroAs property

{button ,AL(`H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_DISPLAYZEROAS_PROPERTY_EXSCRIPT;H_123_ISZERODISPLAYED_PROPERTY_EXSCRIPT',1)} See example

Sets or returns how zeros will be displayed in the specified sheet.

Data type

String

Syntax

sheet.**DisplayZeroAs =** *value*

value **=** *sheet*.**DisplayZeroAs**

Legal values

The value of the DisplayZeroAs property is set in the Sheet - Sheet Properties (#) infobox.

The legal values are multi-character strings.

```
' Example: Contents, DefaultColumnWidth, DefaultRowHeight, DisplayZeroAs,
' IsZeroDisplayed, and Name properties
' This example creates a new sheet and specifies row height and column width settings,
' as well as how to display zeros in the sheet.
Sub SetUpSheet

   ' Create a new sheet.
   Dim mysheet As Sheet
   Set mysheet = CurrentDocument.Newsheet($After,1,True)
   mysheet.Name = "Test Settings"

   ' Specify not to use the font size for the row height,
   ' and set the row height to 12 points.
   mysheet.RowHeightUseFontSize = False
   mysheet.DefaultRowHeight = 12

   ' Set the column width to 14 points.
   mysheet.DefaultColumnWidth = 14

   ' Add a couple of zeros in the sheet.
   [Test Settings:A1].Contents = "0"
   [Test Settings:A2].Contents = "0"

   ' Specify to display zeros as "(empty)".
   mysheet.IsZeroDisplayed = True
   mysheet.DisplayZeroAs = "(empty)"

End Sub
```

1-2-3: Documents property

{button ,AL(`H_123_APPLICATION_CLASS;',0)} See list of classes

{button ,AL(`H_123_DOCUMENTS_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns a list of the files that are open in the specified application.

Data type

Documents

Syntax

Set *documents* **=** *application***.Documents**

Legal values

The value of the Documents property is the Documents object, a collection of files that are open in the specified application.

```
' Example: Count, Documents, and Name properties; Item method
' This example puts the name of all open documents in a string, and
' displays that string.
Sub ShowDocs
   Dim mytext As String
   mytext = "Open documents: "

   ' Loop through all the open documents, and append the document
   ' names to mytext.
   For i = 0 To (CurrentApplication.Documents.Count - 1)
     mytext = mytext + CurrentApplication.Documents.Item(i).Name + " "
   Next
   Messagebox mytext

End Sub
```

1-2-3: Document property

{button ,AL(`H_123_DOCWINDOW_CLASS;',0)} See list of classes

Returns the document object to which the current window belongs.

Data type

Document

Syntax

Set *document* **=** *docwindow*.**Document**

Legal values

The value of the Document property is the Document object to which the document window belongs.

1-2-3: DocWindows property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_DOCWINDOWS_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns a list of all the sheet windows associated with the specified file.

Data type

DocWindows

Syntax

Set *docwindows* **=** *document*.**DocWindows**

Legal values

The value of the DocWindows property is the collection of all the sheet windows associated with the specified file.

```
' Example: Count and DocWindows properties
Sub DocWins
' This example counts the number of open windows in the current document.
' For example, if you have the Preview window and the current sheet showing,
' then this sub shows 2 windows.
   Messagebox "Number of open windows: " + Str(CurrentDocument.DocWindows.Count)
End Sub
```

1-2-3: DoubleUnderline property

{button ,AL(`H_123_FONT_CLASS;',0)} See list of classes

{button ,AL(`H_123_DOUBLEUNDERLINE_PROPERTY_EXSCRIPT;',1)} See example

Determines whether data is styled using the double underline attribute or not.

Data type

Variant (Boolean)

Syntax

font.**DoubleUnderline =** *value*

value **=** *font*.**DoubleUnderline**

Legal values

Value	Description
TRUE	Apply the double underline attribute to the data.
FALSE	(Default) Do not apply the double underline attribute to the data.

```
' Example: Contents, DoubleUnderline, Font, FontName, and Size properties; Select
method
' This example creates a heading with the text "Summary" and then
' specifies text styles for the heading.
Sub CreateHeading
   [a:a4].Select
   Selection.Contents = "Summary"
   Selection.Font.DoubleUnderline = True
   Selection.Font.FontName = "Times New Roman"
   Selection.Font.Size = 14
End Sub
```

1-2-3: DragAndDropEnabled property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether drag and drop is enabled.

Data type

Variant (Boolean)

Syntax

application.**DragAndDropEnabled =** *value*

value **=** *application*.**DragAndDropEnabled**

Legal values

Value	Description
TRUE	(Default) Enable drag and drop.
FALSE	Disable drag and drop.

1-2-3: DrawnObjects property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

(Read-only) Returns an object that is a collection of graphic objects.

Data type

DrawObjects

Syntax

drawobjects = *document*.**DrawnObjects**

Legal values

The following object types are included in the collection returned by the DrawnObjects property: Arc, Ellipse, Drawline (includes arrows), Rectangle (includes rounded rectangles), Polyline, Polygon, Freehand, ButtonControl, Picture, and , EditText.

1-2-3: EdgeColor property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_PLOT_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

{button ,AL(`H_123_EDGECOLOR_PROPERTY_EXSCRIPT;',1)} See example

Sets and returns the color of the border of the specified object.

Data type

Color

Syntax

Set *object.***EdgeColor** = *color*

Set *color* = *object***.EdgeColor**

Legal values

For a list of legal values, see the Color palette.

```
' Example: BackColor, Background, ColorName, EdgeColor, and Pattern properties
Sub MakeBlueRect

   Dim myrect As rectangle

   ' Create a new rectangle
   Set myrect = [A].NewRectangle(1400,1085,3135,1820)

   ' Set the styles of the rectangle
   myrect.EdgeColor.ColorName = "blueberry"
   myrect.EdgeLineWidth = $Thin
   myrect.Background.BackColor.ColorName = "arctic blue"
   myrect.Background.Pattern = $SolidBackground
End Sub
```

1-2-3: EdgeDashStyle property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

Sets or returns the border style for a graphic object or the line style.

Data type

Variant (LineStyleType enumeration)

Syntax

object.**EdgeDashStyle =** *value*

value **=** *object*.**EdgeDashStyle**

Legal values

Value	Description
$None	none
$Solid	———
$LongDash	– —
$Dash	- - - -
$DashDot	— – —
$DashDotDot	·—·–
$Dot	·····
$DotDot	·····

1-2-3: EdgeLineWidth property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

Sets or returns the width of the border of a graphic object or of a line.

Data type

Variant (LineWidths enumeration)

Syntax

object.**EdgeLineWidth =** *value*

value **=** *object*.**EdgeLineWidth**

Legal values

Value	Description
$onepixel	
$verythin	
$thin	
$moderatelythin	
$medium	
$moderatelythick	
$thick	
$verythick	

1-2-3: EditingTime property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_EDITINGTIME_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the total amount of time a file has been open.

Data type

Long

Syntax

value **=** *document*.**EditingTime**

Legal values

The value of the EditingTime property is the total number of minutes that a file has been open. The maximum value is 1092 hours and 15 minutes.

```
' Example: Contents, Description, EditingTime, LastEditor, and Name properties
Sub WriteStatistics
   Dim mysheet As Sheet

   ' Create a new sheet at the end of the document, and name it "Statistics".
   Set mysheet = CurrentDocument.NewSheet($Last,1,True)
   mysheet.Name = "Statistics"

   ' Put the document statistics on the new sheet.
   [Statistics:a1].Contents = "Document contents: " + CurrentDocument.Description
   [Statistics:a2].Contents = "Editing time: " + Str(CurrentDocument.EditingTime) + "
   minutes"
   [Statistics:a3].Contents = "Last editor: " + CurrentDocument.LastEditor
End Sub
```

1-2-3: EditLineVisible property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Determines whether to display the edit line in the application window.

Data type

Variant (Boolean)

Syntax

applicationwindow.**EditLineVisible =** *value*

value **=** *applicationwindow*.**EditLineVisible**

Legal values

Value	Description
TRUE	(Default) Display the edit line in the application window.
FALSE	Do not display the edit line in the application window.

1-2-3: EditPoints property

{button ,AL(`;H_123_DRAWLINE_CLASS;H_123_FREEHAND_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS',0)} See list of classes

Determines whether to display the handles on a specified draw object, so you can move and reshape the object.

Data type

Variant (Boolean)

Syntax

object.**EditPoints** = *value*

Legal values

Value	Description
TRUE	Display the handles on a selected object.
FALSE	Do not display the handles on a selected object.

1-2-3: EmbeddedParticipation property

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS;',0)} See list of classes

(Read-only) Returns the type of menu displayed during an OLE session. User-defined objects can be read-write.

Data type

Variant (EmbeddedParticipationType enumeration)

Syntax

value **=** *object*.**EmbeddedParticipation**

object.**EmbeddedParticipation =** *value*

Legal values

The values for the EmbeddedParticipation property are: $NoGroup, $FileGroup, $EditGroup, $ContainerGroup, $ObjectGroup, $WindowGroup, and $HelpGroup.

1-2-3: Embedded property

{button ,AL(`H_123_DOCUMENT_CLASS;',0)} See list of classes

{button ,AL(`H_123_EMBEDDED_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Determines whether the file is embedded or not.

Data type

Variant (Boolean)

Syntax

value **=** *document*.**Embedded**

Legal values

Value	Description
TRUE	The file is embedded.
FALSE	(Default) The file is not embedded.

```
' Example: Embedded, SynchScrolling, and ViewSplitStyle methods
Sub MySplit
   ' You can split the window only if the workbook isn't embedded in another product.
   If (CurrentDocument.Embedded) Then
       Messagebox "You cannot split the window because the workbook is embedded"
   Else
       CurrentDocument.ViewSplitStyle = $Horizontal
       CurrentDocument.SynchScrolling = True
   End If

End Sub
```

1-2-3: EndColumn

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_STARTROWCOL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the column number for the bottom right cell of the specified range.

Data type

Long

Syntax

value **=** *range***.EndColumn**

Legal values

The value for the EndColumn property is any long from 0 - 255, which correspond to columns A - IV.

1-2-3: EndRow property

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_STARTROWCOL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the row number for the last row of the specified range.

Data type

Long

Syntax

value **=** *range***.EndRow**

Legal values

The value for the EndRow property is any long from 0 - 8,191, which corresponds to row numbers 1 - 8192.

1-2-3: EndSheet property

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_STARTROWCOL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the sheet number of the last sheet of the specified range.

Data type

Long

Syntax

value **=** *range***.EndSheet**

Legal values

The value for the EndSheet property is an long from 0 - 255 that corresponds to the sheet number.

1-2-3: EveningString property

{button ,AL(`H_123_APPLICATION_CLASS;',0)} See list of classes

(Read-only) Returns a string for the PM symbol used to indicate evening time.

Data type

String

Syntax

value **=** *application*.**EveningString**

Legal values

The default value of the EveningString property is PM. You set the value of the EveningString property in the Windows Control Panel.

1-2-3: Events property

{button ,AL(`H_123_CLASSINFO_CLASS;',0)} See list of classes

(Read-only) Returns a collection of the events that can be raised by a class.

Data type

Strings

Syntax

strings **=** *classinfo***.Events**

Legal values

The value of the Events property is a collection of strings containing the names of the events for the class.

1-2-3: ExtractingUniqueRecords property

{button ,AL(`H_123_QUERY_CLASS;',0)} See list of classes

Determines whether a query should return unique records only.

Data type

Variant (Boolean)

Syntax

dataquery.**ExtractingUniqueRecords =** *value*

value **=** *dataquery*.**ExtractingUniqueRecords**

Legal values

Value	Description
TRUE	(Default) Return unique records only.
FALSE	Do not return unique records only.

1-2-3: DataQueryNames property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SQL_PROPERTY_EXSCRIPT;H_123_QUERYNAMES_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns a collection of the existing query names in the specified file.

Data type

Strings

Syntax

Set *strings* **=** *document*.**DataQueryNames**

Legal values

The value for the DataQueryNames property is a collection of the existing query names.

```
' Example: Contents and DataQueryNames properties; MoveCellPointer and OpenDocument
methods
' List in column A the current names of all queries in a file.
' You must change the file name in the line below to refer to
' the file you want the query to run against.
Const QRYFILE = "c:\lotus\work\123\qtfile.123"
Dim myfile As Document
Dim qnames As Strings
' Open a file containing queries
Set myfile = CurrentApplication.OpenDocument(QRYFILE, "", "", "", True)
' Get all query names
Set qnames = myfile.DataQueryNames
' Move to cell A1
.MoveCellPointer $Home, 1
' Show all query names in column A
Forall X In QNames
' Put name in cell
Selection.Contents = X
' Move down 1 row
.MoveCellPointer $Down, 1
End Forall
```

1-2-3: IsBubbleHelp property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Determines whether the bubble help is displayed.

Data type

Variant (Boolean)

Syntax

applicationwindow.**IsBubbleHelp =** *value*

value **=** *applicationwindow*.**IsBubbleHelp**

Legal values

Value	Description
TRUE	(Default) Display bubble help.
FALSE	Do not display bubble help.

1-2-3: FitDrawnObjectToPage property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the type of scaling to be done for the printed output of graphic objects.

Data type

Variant (FitDrawToPage enumeration)

Syntax

printsettings.**FitDrawnObjectToPage** = *value*

value **=** *printsettings*.**FitDrawnObjectToPage**

Legal values

Value	Description
$FillNone	No scaling.
$FillPage	Fit the graphic to the page.
$FillPageProportions	Fit the graphic to the page while maintaining proportions.
$FillCustom	Use a specified scale to fit the graphic to the page.

1-2-3: FitRowHeightToFont property

{button ,AL(`;H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether the row height in a new file automatically fits entries with the largest font or not.

Data type

Variant (Boolean)

Syntax

application.**FitRowHeightToFont =** *value*

value **=** *application*.**FitRowHeightToFont**

Legal values

Value	Description
TRUE	(Default) Automatically fit entries with the largest font.
FALSE	Do not automatically fit entries with the largest font.

1-2-3: FitToPage property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the page fit for the print selection.

Data type

Variant (Scaling enumeration)

Syntax

printsettings.**FitToPage =** *value*

value **=** *printsettings*.**FitToPage**

Legal values

Value	Description
$None	Print the actual size.
$FitRows	Fit all rows on the page.
$FitColumns	Fit all columns on the page.
$FitRowsAndColumns	Fit all rows and columns on the page.
$CustomFit	Manually scale the print range.

1-2-3: FontColor property

{button ,AL(`H_123_FONT_CLASS',0)} See list of classes

{button ,AL(`H_123_FONTCOLOR_PROPERTY_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT;H_123_PATTERN_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the color of the font for an object.

Data type

Color

Syntax

Set *font***.FontColor =** *color*

Set *color* **=** *font***.FontColor**

Legal values

The value of the FontColor is the color object for the font for an object.

Usage

The FontColor property applies to any object that can have font and text styling.

```
' Example: Cells, CellValue, ColorName, Font, andFontColor properties
' Assigns a color to each value in a range, depending on if the value is
' larger or smaller than a specified value.
Sub SalesQuota
   'This script acts on each cell in a selected range.
   ForAll x In Selection.Cells
      'Color all values greater than 500 blue.
      If x.CellValue > 500 Then
         x.Font.FontColor.ColorName = "blue"
      Else
         'Color all values less than 500 red.
         x.Font.FontColor.ColorName = "red"
      End If
   End Forall
End Sub
```

1-2-3: FontName property

{button ,AL(`H_123_FONT_CLASS',0)} See list of classes

{button ,AL(`H_123_FONTNAME_PROPERTY_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT;H_123_FONT_PROPERTY_EXSCRIPT `,1)} See example

Sets or returns the name of the font for an object.

Data type

String

Syntax

font.**FontName =** *value*

value **=** *font*.**FontName**

Legal values

The value of the FontName property is a string containing the font name.

Usage

The FontName property applies to any object that can have font and text styling. The available font names depend on what you have installed on your system.

```
' Example: Font, FontName, and Size properties
Sub ChangeFont
   'Set the font name and font size for the currently selected range.
   Selection.Font.FontName = "Arial Narrow"
   Selection.Font.Size = 12
End Sub
```

1-2-3: Font property

{button ,AL(`H_123_BUTTONCONTROL_CLASS;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_MAPTITLE_CLASS;',0)} See list of classes

{button ,AL(`H_123_FONT_PROPERTY_EXSCRIPT;H_123_STRIKETHROUGH_PROPERTY_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT;H_123_PATTERN_PROPERTY_EXSCRIPT;H_123_FONTCOLOR_PROPERTY_EXSCRIPT;H_123_FONTNAME_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns the font and text styles of the object.

Data type

Font

Syntax

Set *font* **=** *object*.**Font**

Legal values

The values of the Font property are the font and text styles of the object.

Usage

The Font property applies to text and numbers in buttons, text blocks, query tables, and ranges.

```
' Example: Font, FontName, RangeSelector, and Size properties
' Select a range and then set text styles for the range.
Sub FontStyle
   ' The RangeSelector object lets the user select a range during script execution.
   Dim rs As RangeSelector
   Set rs = CurrentApplication.RangeSelector
   Dim r As Range
   Set r = rs.GetRange
   ' Change the font in the selected range to Arial Narrow, 14 point.
   r.Font.FontName = "Arial Narrow"
   r.Font.Size = 14
End Sub
```

1-2-3: FooterCenterFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the font for the center footer of the print selection.

Data type

Font

Syntax

Set *printsettings*.**FooterCenterFont =** *font*

Set *font* **=** *printsettings*.**FooterCenterFont**

Legal values

The value of the FooterCenterFont property is the font for the center footer, set in the Preview & Page Setup InfoBox.

1-2-3: FooterCenter property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the text of the center footer of the print selection.

Data type

String

Syntax

printsettings.**FooterCenter =** *value*

value **=** *printsettings*.**FooterCenter**

Legal values

The value of the FooterCenter property can include the current date, the current time, the page number, the name of the file, and the cell contents. To include these items, use the symbols shown in the following table.

Value	Description
@ (at sign)	Date of printing
+ (plus sign)	Time of printing
# (pound sign)	Page number
% (percent sign)	Total number of pages
^ (caret)	File name
\ (backslash) followed by a cell address or range name	Contents of a cell

1-2-3: FooterLeftFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the font for the left footer of the print selection.

Data type

Font

Syntax

Set *printsettings*.**FooterLeftFont =** *font*

Set *font* **=** *printsettings*.**FooterLeftFont**

Legal values

The value of the FooterLeftFont property depends on the fonts you have installed on your computer.

1-2-3: FooterLeft property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the text of the left footer of the print selection.

Data type

String

Syntax

printsettings.**FooterLeft =** *value*

value **=** *printsettings*.**FooterLeft**

Legal values

The value of the FooterLeft property can include the current date, the current time, the page number, the name of the file, and the cell contents. To include these items, use the symbols shown in the following table.

Value	Description
@ (at sign)	Date of printing
+ (plus sign)	Time of printing
# (pound sign)	Page number
% (percent sign)	Total number of pages
^ (caret)	File name
\ (backslash) followed by a cell address or range name	Contents of a cell

1-2-3: FooterRightFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the font for the right footer of the print selection.

Data type

Font

Syntax

Set *printsettings*.**FooterRightFont =** *font*

Set *font* **=** *printsettings*.**FooterRightFont**

Legal values

The value of the FooterRightFont property depends on the fonts you have installed on your computer.

1-2-3: FooterRight property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the text of the right footer of the print selection.

Data type

String

Syntax

printsettings.**FooterRight =** *value*

value **=** *printsettings*.**FooterRight**

Legal values

The value of the FooterRight property can include the current date, the current time, the page number, the name of the file, and the cell contents. To include these items, use the symbols shown in the following table.

Value	Description
@ (at sign)	Date of printing
+ (plus sign)	Time of printing
# (pound sign)	Page number
% (percent sign)	Total number of pages
^ (caret)	File name
\ (backslash) followed by a cell address or range name	Contents of a cell

```
' Example: FormatDecimals and FormatName properties; FitWidest method
' This example formats the values in the current range as
' Spanish Peseta with 0 decimal places.
Sub MyNumbers
   ' Set currency format with 0 decimal places.
   Selection.FormatName = "Spanish Peseta"
   Selection.FormatDecimals = 0
   ' Make sure column is wide enough to accommodate new format.
   Selection.FitWidest
End Sub
```

1-2-3: FormatDecimals property

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_FORMATDECIMALS_AND_FORMATNAME_PROPERTIES_EXSCRIPT;H_123_NEGATIVESIN COLOR_PROPERTY_EXSCRIPT;H_123_ISPARENTHESIZED_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the number of digits displayed after the decimal point for the specified range or sheet.

Data type

Long

Syntax

object.**FormatDecimals =** *value*

value **=** *object*.**FormatDecimals**

Legal values

The value of the FormatDecimals property is any long from 0-15. The default value is what you specified in the Windows Control Panel.

1-2-3: FormatName property

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_FORMATDECIMALS_AND_FORMATNAME_PROPERTIES_EXSCRIPT;H_123_NEGATIVESIN COLOR_PROPERTY_EXSCRIPT;H_123_ISPARENTHESIZED_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the number format for a range or sheet.

Data type

Variant (String)

Syntax

object.**FormatName =** *value*

value **=** *object*.**FormatName**

Legal values

The values for the FormatName property are the currency, date, time, and number formats listed in the Range - Range Properties and Sheet - Sheet Properties infoboxes.

1-2-3: FormatProtected property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_FORMATPROTECTED_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Determines whether styles in a file are protected or not.

Data type

Variant (Boolean)

Syntax

value **=** *document*.**FormatProtected**

Legal values

Value	Description
TRUE	Styles are locked.
FALSE	(Default) Styles are not locked.

```
' Example: FormatProtected property
' This example tests to see if the styles for the current
' document are protected.
Sub ChangeStyle
   ' Check to see if the styles are protected.
   If CurrentDocument.FormatProtected Then
      'If they are, display this message.
      Messagebox "This workbook is locked. You cannot change styles."
   Else
      'If they aren't, display this message.
      Messagebox "This workbook is not locked. You can change styles."
   End If
End Sub
```

1-2-3: Format property

{button ,AL(`;H_123_DATALINK_CLASS',0)} See list of classes

(Read-only) Returns the Clipboard format that is associated with a Datalink object.

Data type

Variant (ClipboardFormat enumeration)

Syntax

value **=** *datalink*.**Format**

Legal values

The values for the Format property are: $TextFormat, $Wk1Format, and $Wk3Format.

1-2-3: FormulaFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the text style object for formulas in the print selection.

Data type

Font

Syntax

Set *printsettings*.**FormulaFont =** *font*

Set *font* **=** *printsettings*.**FormulaFont**

Legal values

The value of the FormulaFont property is a font object.

1-2-3: FormulasPrint property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether formulas in the print selection will print the text of the formula on the following pages.

Data type

Variant (Boolean)

Syntax

printsettings.**FormulasPrint =** *value*

value **=** *printsettings*.**FormulasPrint**

Legal values

Value	Description
TRUE	Print formulas.
FALSE	(Default) Do not print formulas.

1-2-3: FrameColor property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_MAP_CLASS;H_123_MAPPLOT_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

Sets or returns the color of an object's frame.

Data type

Color

Syntax

Set *object*.**FrameColor =** *color*

Set *color* **=** *object*.**FrameColor**

Legal values

The value for the FrameColor property is a color object.

1-2-3: FullName property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the full path name of the application's executable file.

Data type

String

Syntax

value **=** *application*.**FullName**

Legal values

The value of the FullName property is a string containing the full path (drive, directory, and applicable subdirectories) in which the application's executable file is located.

1-2-3: GMTTime property

{button ,AL(`H_123_DATETIME_CLASS',0)} See list of classes

(Read-only) Returns the date and time, in Greenwich mean time.

Data type

String

Syntax

value **=** *datetime*.**GMTTime**

Legal values

The value for the GMTTime property can be any date/time string.

1-2-3: Green property

{button ,AL(`;H_123_COLOR_CLASS',0)} See list of classes

{button ,AL(`H_123_GREEN_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns the green component of the Color object which is contained by any object that can be styled, such as a border or a text block.

Data type

Long

Syntax

value **=** *color*.**Green**

Legal values

The value of the Green property is any long from 0 - 255.

```
' Example: BackColor, Background, Contents, Green, and Name properties
' This example enters information about a color in the sheet.
Sub AmountRedGreen
   Cn = [A1].Background.BackColor.Name
   Gr = [A1].Background.BackColor.Green
   Rd = [A1].Background.BackColor.Red
   'Enter the name of the the background color of cell A1.
   [B1].Contents = "Color: "&cn
   'Enter the amount of green in the background color of cell A1.
   [B2].Contents = "Green: "&gr
   'Enter the amount of red in the background color of cell A1.
   [B3].Contents = "Red: "&rd
End Sub
```

1-2-3: GridBorder property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_gridborder_PROPERTY_EXSCRIPT;',1)} See example

With the Style or Color property, sets or returns the style or color of the grid border of a range.

Data type

RangeBorder

Syntax

Set *range*.**GridBorder.Style =** *value*

Set *range*.**GridBorder.Color =** *value*

Set *range*.**GridBorder =** *rangeborder*

Set *value* **=** *range*.**GridBorder.Style**

Set *value* **=** *range*.**GridBorder.Color**

Set *rangeborder* **=** *range*.**GridBorder**

Legal values

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

A RangeBorder object.

```
' Example: Color, ColorName, and GridBorder properties
' Add a gray, double-line grid border to a range of values.
Sub OutlineCells
   Selection.GridBorder.Style = $DoubleBorder
   Selection.GridBorder.Color.ColorName = "50% gray"
End Sub
```

1-2-3: GridLineColor property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_GRIDLINECOLOR_PROPERTY_EXSCRIPT;H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the color of grid lines in the specified sheet, file, or application.

Data type

Color

Syntax

Set *object*.**GridLineColor =** *color*

Set *color* **=** *object*.**GridLineColor**

Legal values

The value of the GridLineColor property is a Color object.

```
' Example: GridLineColor, ShowAutomaticPageBreaks, ShowGridLines,
ShowManualPageBreaks, ShowSheetFrame, and ZoomScale properties
' Sets the following view preferences: Hides the sheet frame,
' displays green grid lines, hides both manual and automatic page breaks,
' and sets the zoom scale to 90%.
Sub ViewPrefs
   CurrentDocument.ShowSheetFrame = False
   CurrentDocument.ShowGridLines = True
   CurrentDocument.GridLineColor.ColorName = "green"
   CurrentDocument.ShowManualPageBreaks = False
   CurrentDocument.ShowAutomaticPageBreaks = False
   CurrentDocument.ZoomScale = 90
End Sub
```

1-2-3: GridLinesPrint property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether grid lines are printed with the print selection.

Data type

Variant (Boolean)

Syntax

printsettings.**GridLinesPrint =** *value*

value **=** *printsettings*.**GridLinesPrint**

Legal values

Value	Description
TRUE	Print data with grid lines.
FALSE	Do not print grid lines with the data.

1-2-3: HasPassword property

{button ,AL(`;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_HASPASSWORD_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Determines whether the file is protected with a password.

Data type

Variant (Boolean)

Syntax

value **=** *document***.HasPassword**

Legal values

Value	Description
TRUE	The file is protected with a password.
FALSE	(Default) The file is not protected with a password.

```
' Example: HasPassword property
' Make the user specify a password before saving a file.
' The PreSave event function checks to see if the file has been saved with a
' password and blocks the save if it has not. To ensure that the user saves
' files with a password, attach these scripts to the PreSaveAs and
' PostSaveAs events, as well.
Function PreSave(Source As Document) As Variant
   If CurrentDocument.HasPassword Then
      ' User specified password.
      ' Continue the Save.
      PreSave = $Continue
   Else
      ' User did not specify a password.
      ' Block the Save and raise PostSave.
      PreSave = $Block
   End If
End Function

Sub PostSave(Source As Document, P1 As Variant)
   If P1 = $Block Then
      Messagebox "You must specify a password when you save this file.",,"1-2-3"
   Else
      Messagebox "The workbook was saved.",,"1-2-3"
   End If
End Sub
```

1-2-3: HeaderCenterFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the font for the center header of the print setting.

Data type

Font

Syntax

Set *printsettings*.**HeaderCenterFont =** *font*

Set *font* **=** *printsettings*.**HeaderCenterFont**

Legal values

The value of the HeaderCenterFont property depends on the fonts you have installed on your computer.

1-2-3: HeaderCenter property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the text of the center header of the print setting.

Data type

String

Syntax

printsettings.**HeaderCenter =** *value*

value **=** *printsettings*.**HeaderCenter**

Legal values

The value of the HeaderCenter property can include any text. In addition, it can include the current date, the current time, the page number, the total number of pages, the name of the file, and the contents of a cell. To include these items, use the symbols shown in the following table.

Value	Description
@ (at sign)	Date of printing
+ (plus sign)	Time of printing
# (pound sign)	Page number
% (percent sign)	Total number of pages
^ (caret)	File name
\ (backslash) followed by a cell address or range name	Contents of a cell

1-2-3: HeaderLeftFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the font for the left header of the print setting.

Data type

Font

Syntax

Set *printsettings*.**HeaderLeftFont =** *font*

Set *font* **=** *printsettings*.**HeaderLeftFont**

Legal values

The value of the HeaderLeftFont property depends on the fonts you have installed on your computer.

1-2-3: HeaderLeft property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the text of the left header of the print setting.

Data type

String

Syntax

printsettings.**HeaderLeft =** *value*

value **=** *printsettings*.**HeaderLeft**

Legal values

The value of the HeaderLeft property can include any text. In addition, it can include the current date, the current time, the page number, the name of the file, and the cell contents. To include these items, use the symbols shown in the following table.

Value	Description
@ (at sign)	Date of printing
+ (plus sign)	Time of printing
# (pound sign)	Page number
% (percent sign)	Total number of pages
^ (caret)	File name
\ (backslash) followed by a cell address or range name	Contents of a cell

1-2-3: HeaderRightFont property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the font for the right header of the print setting.

Data type

Font

Syntax

Set *printsettings*.**HeaderRightFont =** *font*

Set *font* **=** *printsettings*.**HeaderRightFont**

Legal values

The value of the HeaderRightFont property depends on the fonts you have installed on your computer.

1-2-3: HeaderRight property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the text of the right header of the print setting.

Data type

String

Syntax

printsettings.**HeaderRight =** *value*

value **=** *printsettings*.**HeaderRight**

Legal values

The value of the HeaderRight property can include any text. In addition, it can include the current date, the current time, the page number, the name of the file, and the cell content. To include these items, use the symbols shown in the following table.

Value	Description
@ (at sign)	Date of printing
+ (plus sign)	Time of printing
# (pound sign)	Page number
% (percent sign)	Total number of pages
^ (caret)	File name
\ (backslash) followed by a cell address or range name	Contents of a cell

1-2-3: Height property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS',0)} See list of classes

{button ,AL(`H_123_HEIGHT_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the height of the window, or the height of the bounding box around a graphic object.

Data type

Long

Syntax

Set *object*.**Height =** *value*

Set *value* **=** *object*.**Height**

Legal values

For window objects, the value of the Height property is the height of the window, in units of pixels. For graphic objects, it is the height of the object's bounding box, in units of twips.

Usage

If the window is minimized or maximized, you will not see the effect of the Height property setting until the window is restored.

```
' Example: Height property
' Set the height of all charts in a workbook.
Sub SizeCharts
   ' This script acts on all charts in the current workbook.
   ForAll x in CurrentDocument.Charts
   ' Set the height of all charts to 2000 pixels.
   x.Height = 2000
End ForAll
End Sub
```

1-2-3: HorizontalBorder property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

With the Style or Color property, sets or returns the style or color of the horizontal border of a range.

Data type

RangeBorder

Syntax

Set *range*.**HorizontalBorder =** *rangeborder*

range.**HorizontalBorder.Style =** *value*

Set *range*.**HorizontalBorder.Color =** *color*

Set *rangeborder* **=** *range*.**HorizontalBorder**

value **=** *range*.**HorizontalBorder.Style**

Set *color* **=** *range*.**HorizontalBorder.Color**

Legal values

A RangeBorder object.

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

```
' Example: Color, ColorName, and HorizontalBorder properties
' Add a horizontal border between the rows of a range.
Sub HBorder
   Selection.HorizontalBorder.Style = $DoubleBorder
   Selection.HorizontalBorder.Color.ColorName = "light burgundy"
End Sub
```

```
' Example: HorizontalPageBreak and VerticalPageBreak properties
' Set a horizontal page break above row 20 and to the left
' of column E on the sheet named Budget.
Sub MakePage
   [Budget:E20].HorizontalPageBreak = True
   [Budget:E20].VerticalPageBreak = True
End Sub

' Delete a horizontal page break above row 20 and to the left
' of column E on the sheet named Budget.
Sub NoBreaks
   [Budget:E20].HorizontalPageBreak = False
   [Budget:E20].VerticalPageBreak = False
End Sub
```

1-2-3: HorizontalPageBreak property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_HORIZONTALPAGEBREAK_AND_VERTICALPAGEBREAK_PROPERTIES_EXSCRIPT',1)} See example

Determines whether to insert or delete a horizontal page break above the topmost row in the specified range.

Data type

Variant (Boolean)

Syntax

range.**HorizontalPageBreak =** *value*

value **=** *range*.**HorizontalPageBreak**

Legal values

Value	Description
TRUE	Insert a horizontal page break.
FALSE	Delete the horizontal page break.

1-2-3: HorizontalScrollBarVisible property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)}
See list of classes

(Read-only) Returns whether to display the horizontal scroll bar in the window or not.

Data type

Variant (Boolean)

Syntax

value **=** *object*.**HorizontalScrollBarVisible**

Legal values

Value	Description
TRUE	(Default) Display the horizontal scroll bar.
FALSE	Hide the horizontal scroll bar.

1-2-3: HorizontalTitle property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_HORIZONTALTITLE_PROPERTY_EXSCRIPT ',1)} See example

Determines whether the horizontal titles are frozen, based on the current location of the cell pointer.

Data type

Variant (Boolean)

Syntax

sheet.**HorizontalTitle =** *value*

value **=** *sheet*.**HorizontalTitle**

Legal values

Value	Description
TRUE	Freeze horizontal titles.
FALSE	(Default) Do not freeze horizontal titles.

```
' Example: HorizontalTitle and VerticalTitle properties
' Moves the cell pointer to A:B2 and freezes row 1 and column A as titles.
Sub MakeTitles
   [A:B2].Select
   [A].HorizontalTitle = True
   [A].VerticalTitle = True
End Sub
```

1-2-3: IconBarNames property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

(Read-only) Returns a collection of the names of the available sets of SmartIcons, including the names of custom sets of SmartIcons you created.

Data type

Strings

Syntax

Set *strings* **=** *applicationwindow*.**IconBarNames**

Legal values

The value of the IconBarNames property is a collection of strings containing the names of the sets of SmartIcons available in 1-2-3, including custom sets.

1-2-3: IconBarsVisible property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Determines whether all sets of SmartIcons are displayed.

Data type

Variant (Boolean)

Syntax

applicationwindow.**IconBarsVisible =** *value*

value **=** *applicationwindow*.**IconBarsVisible**

Legal values

Value	Description
TRUE	Display the sets of SmartIcons.
FALSE	Hide the sets of SmartIcons.

1-2-3: IconSize property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Sets or returns the display size of icons.

Data type

Variant (IconSizeType enumeration)

Syntax

applicationwindow.**IconSize** = *value*

value = *applicationwindow*.**IconSize**

Legal values

Value	Description
$Regular	(Default) Display small-sized icons.
$Large	Display large-sized icons.

```
' Example: InitialColWidth and InitialRowHeight properties
' Set the initial column width and row height for new workbooks
' and then create a new workbook.
Dim document1 as Document
' Set the initial row height and column width for new workbooks.
CurrentApplication.FitRowHeightToFont = False
CurrentApplication.InitialColWidth = 15
CurrentApplication.InitialRowHeight = 72
' Create a new workbook called MyDoc.
Set document1 = CurrentApplication.NewDocument("MyDoc", "", "", "", "",,)
```

1-2-3: InitialColWidth property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_INITIALCOLWIDTH_AND_INITIALROWHEIGHT_PROPERTIES_EXSCRIPT ',1)} See example

Sets or returns the column width for a new file.

Data type

Long

Syntax

application.**InitialColWidth =** *value*

value **=** *application*.**InitialColWidth**

Legal values

The value of the InitialColWidth property is any integer from 1 - 240. 1-2-3 sizes columns in whole-character increments; the default is 9 characters.

1-2-3: InitialRowHeight property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_INITIALCOLWIDTH_AND_INITIALROWHEIGHT_PROPERTIES_EXSCRIPT ',1)} See example

Sets or returns the row height for a new file.

Data type

Long

Syntax

application.**InitialRowHeight =** *value*

value **=** *application*.**InitialRowHeight**

Legal values

The value of the InitialRowHeight property is any integer from 1 - 255. Row height is set in points. The default value is 14.

1-2-3: InnerBorder property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_INNERBORDER_PROPERTY_EXSCRIPT ',1)} See example

With the Style or Color property, sets or returns the style or color of the inner border of a range.

Data type

RangeBorder

Syntax

Set *range*.**InnerBorder =** *rangeborder*

range.**InnerBorder.Style =** *value*

Set *range*.**InnerBorder.Color =** *color*

Set *rangeborder* **=** *range*.**InnerBorder**

value **=** *range*.**InnerBorder.Style**

Set *color* **=** *range*.**InnerBorder.Color**

Legal values

A RangeBorder object.

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

```
' Example: Color; ColorName; and InnerBorder properties
' Add a solid, turquoise inner border to the selected range.
Sub MakeBorder
   Selection.InnerBorder.Color.ColorName = "turquoise"
   Selection.InnerBorder.Style = $SolidBorder
End Sub
```

1-2-3: InsidePlot property

{button ,AL(`H_123_LEGEND_CLASS',0)} See list of classes

Determines whether to place the legend in the plot area of a map frame or not.

Data type

Variant (Boolean)

Syntax

legend.**InsidePlot =** *value*

value **=** *legend*.**InsidePlot**

Legal values

Value	Description
TRUE	(Default) Place legend in the plot area.
FALSE	Do not place legend in the plot area.

1-2-3: Interactive property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Determines whether 1-2-3 currently has focus and so can accept user input.

Data type

Variant (Boolean)

Syntax

value **=** *application*.**Interactive**

Legal values

Value	Description
TRUE	1-2-3 accepts user input.
FALSE	1-2-3 does not accept user input.

1-2-3: InternetIconsVisible property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Determines whether the set of Internet SmartIcons is displayed or not.

Data type

Variant (Boolean)

Syntax

applicationwindow.**InternetIconsVisible =** *value*

value **=** *applicationwindow*.**InternetIconsVisible**

Legal values

Value	Description
TRUE	Set of Internet SmartIcons is displayed.
FALSE	(Default) Set of Internet SmartIcons is not displayed.

```
' Example: IsColumnCollapsed and IsRowCollapsed properties
Sub Collapse
[A:A1].DemoteColumn 1
[B1].CollapseColumn
[A:A1].DemoteRow 1
[A2].CollapseRow
If [B1].IsColumnCollapsed = True Then
Messagebox "Column Collapsed"
Else
Messagebox "Not Collapsed"
End If
If [A2].IsRowCollapsed = True  Then
Messagebox "Row Collapsed"
Else
Messagebox "Row Not Collapsed"
End If
End Sub
```

1-2-3: IsColumnCollapsed property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ISCOLUMNCOLLAPSED_AND_ISROWCOLLAPSED_PROPERTIES_EXSCRIPT ',1)} See example

(Read-only) Determines whether any of the columns in an outlined range are collapsed.

Data type

Variant (Boolean)

Syntax

value **=** *range*.**IsColumnCollapsed**

Legal values

Value	Description
TRUE	Columns are collapsed.
FALSE	Columns are not collapsed

```
' Example: IsColumnHidden and IsRowHidden properties
' Check a range for hidden columns or rows. Use this code to
' prevent unexpected results in a script that asks the user
' to specify a range to make changes to. (If hidden columns and rows are
' not protected and the sheet or workbook that contains them is not
' locked, scripts that enter new data can write over data in the hidden
' columns and rows.)
Sub OutlineCheck
   Dim r As Range
   Set r = [A:B1..A:B4]
      'If the range contains hidden rows, display this message.
      If r.IsRowHidden = True Then
         MessageBox "Error! Range contains hidden rows."
          'If the range contains hidden columns, display this message.
      ElseIf r.IsColumnHidden = True Then
          MessageBox "Error! Range contains hidden columns."
      'If the specified range contains no hidden rows or columns, display this
   message.
      Else
         MessageBox "Range contains no hidden columns or rows."
      End If
End Sub
```

1-2-3: IsColumnHidden property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ISCOLUMNHIDDEN_AND_ISROWHIDDEN_PROPERTIES_EXSCRIPT ',1)} See example

(Read-only) Determines whether columns in a specified range are hidden.

Data type

Variant (Boolean)

Syntax

value = *range*.**IsColumnHidden**

Legal values

Value	Description
TRUE	Columns are hidden.
FALSE	Columns are displayed.

1-2-3: IsDraggable property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_ARC_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

(Read-only) Determines whether or not an object can be cut or copied to the Clipboard, or dragged and dropped.

Data type

Variant (Boolean)

Syntax

value **=** *object*.**IsDraggable**

Legal values

Value	Description
TRUE	The object can be cut, copied, or dragged and dropped.
FALSE	The object cannot be cut, copied, or dragged and dropped.

1-2-3: IsFormatFreqUsed property

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ISFORMATFREQUSED_PROPERTY_EXSCRIPT ',1)} See example

Determines whether the current format appears in the Frequently Used list.

Data type

Variant (Boolean)

Syntax

object.**IsFormatFreqUsed** = *value*

value = *object*.**IsFormatFreqUsed**

Legal values

Value	Description
TRUE	The format appears in the Frequently Used list.
FALSE	The format does not appear in the Frequently Used list.

```
' Example: IsFormatFreqUsed property
' Promotes the number format of the selected cell to the
' frequently used list in the status bar.
Sub PromoteMyFormat
   ' Check to see if the number format of the current cell is in the frequently used
   list.
   If Selection.IsFormatFreqUsed = False Then
      ' If not, then promote the format.
      Selection.IsFormatFreqUsed = True
   Else
      ' Otherwise, display this message.
      Messagebox "Format is already in the Frequently used list."
   End If
End Sub
```

```
' Example: BackColor, Background, IsHidden, IsProtected, and Pattern  properties
' Hide the data in a specified range, protect the range from changes
' and add a pattern to it.
Sub HideData
   ' Lock Sheet A.
   [A].IsProtected = True
   ' Protect the selected range and hide any data in it.
   Selection.IsHidden = True
   Selection.IsProtected = True
   ' Add a pattern to the range to alert users that it contains hidden data.
   Selection.Background.Pattern = $DoubleRightHatch
   Selection.Background.BackColor.ColorName = "25% gray"
End Sub
```

1-2-3: IsHidden property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ISHIDDEN_AND_ISPROTECTED_PROPERTIES_EXSCRIPT ',1)} See example

Determines whether the range is hidden.

Data type

Variant (Boolean)

Syntax

range.**IsHidden =** *value*

value **=** *range*.**IsHidden**

Legal values

Value	Description
TRUE	Hide the range.
FALSE	(Default) Display the range.

1-2-3: IsLeapYear property

{button ,AL(`H_123_DATETIME_CLASS',0)} See list of classes

(Read-only) Determines whether the date specified by the DateTime object occurs in a leap year.

Data type

Variant (Boolean)

Syntax

value **=** *datetime*.**IsLeapYear**

Legal values

Value	Description
TRUE	The date specified occurs in a leap year.
FALSE	The date specified does not occur in a leap year.

1-2-3: IsLocked property

{button ,AL(`;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS',0)} See list of classes

{button ,AL(`H_123_ISLOCKED_PROPERTY_EXSCRIPT ',1)} See example

Determines whether the graphic object is locked.

Data type

Variant (Boolean)

Syntax

object.**IsLocked =** *value*

value **=** *object*.**IsLocked**

Legal values

Value	Description
TRUE	Lock the graphic object.
FALSE	(Default) Do not lock the graphic object.

Usage

When an object is locked, you cannot move, size, delete, style, or manipulate it.

```
' Example: IsLocked property
' Lock all the OLE objects in the current workbook.
Sub LockObjects
   ForAll x.CurrentDocument.OLEObjects
      x.IsLocked = True
   End ForAll
End Sub
```

1-2-3: IsNew property

{button ,AL(`H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS',0)} See list of classes

(Read-only) Returns whether the current version or version group has been added since you last saved the file.

Data type

Variant (Boolean)

Syntax

value **=** *object***.IsNew**

Legal values

Value	Description
TRUE	Version or version group has been added since the file was last saved.
FALSE	Version or version group has not been added since the file was last saved.

1-2-3: IsNotesFX property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

Determines whether the current range is used for Notes/FX.

Data type

Variant (Boolean)

Syntax

range.**IsNotesFX =** *value*

value **=** *range*.**IsNotesFX**

Legal values

Value	Description
TRUE	Range is used for Notes/FX.
FALSE	Range is not used for Notes/FX.

1-2-3: IsParenthesized property

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ISPARENTHESIZED_PROPERTY_EXSCRIPT ',1)} See example

Determines whether parentheses are placed around all numbers in a range or a sheet.

Data type

Variant (Boolean)

Syntax

object.**IsParenthesized** = *value*

value = *object*.**IsParenthesized**

Legal values

Value	Description
TRUE	Place parentheses around all numeric values.
FALSE	(Default) Do not place parentheses around all numeric values.

```
' Example: FormatDecimals, FormatName, and IsParenthesized properties
' Format values in the selected range as parethesized,
' and fixed with no decimal places.
Sub ParenFormat
   Selection.FormatName = "Fixed"
   Selection.FormatDecimals = 0
   Selection.IsParenthesized = True
End Sub
```

1-2-3: IsProtected property

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ISHIDDEN_AND_ISPROTECTED_PROPERTIES_EXSCRIPT ',1)} See example

Determines whether the current range, table, or sheet is protected.

Data type

Variant (Boolean)

Syntax

object.**IsProtected =** *value*

value **=** *object*.**IsProtected**

Legal values

Value	Description
TRUE	Protect the range, table, or sheet.
FALSE	Do not protect the range, table, or sheet.

The default value of the IsProtected property for a range is True. The default value for a sheet is False.

1-2-3: IsRangeNamed property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ISRANGENAMED_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Determines whether a specified range is named.

Data type

Variant (Boolean)

Syntax

value **=** *range*.**IsRangeNamed**

Legal values

Value	Description
TRUE	Specified range is named.
FALSE	(Default) Specified range is not named.

{button ,AL(`H_123_CLEARRANGENAMES_METHOD_MEMDEF;H_123_CREATERANGENAME_METHOD_MEMDEF;H_123_CREATERANGENAMEFROMLABEL_METHOD_MEMDEF;H_123_CREATERANGENAMETABLE_METHOD_MEMDEF;H_123_NAME_PROPERTY_MEMDEF',0)} See related topics

```
' Example: CoordinateString, IsRangeNamed, and Name properties
' Tell the user if a selected range is named and, if it is,
' what the name is.
Sub AboutRange
   ' Uses RangeSelector object to let the user select a range.
   Dim rs As RangeSelector
   Set rs = CurrentApplication.RangeSelector
   Dim r As Range
   Set r = rs.GetRange
   ' Variable rn is the name of the selected range.
   rn = r.Name
   ' Variable cs is the address of the selected range.
   cs = r.CoordinateString
   Noname = "The range "& cs &" is not named."
   Nameis = "The range "& cs &" is named "& rn
      ' If the selected range is not named, display a message that says so.
      If r.IsRangeNamed = False Then
      Messagebox Noname
         ' If the selected range is named, display the name in a message.
         Else
         Messagebox Nameis
      End If
End Sub
```

1-2-3: IsRowCollapsed property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ISCOLUMNCOLLAPSED_AND_ISROWCOLLAPSED_PROPERTIES_EXSCRIPT ',1)} See example

(Read-only) Determines whether any of the rows in an outlined range are collapsed.

Data type

Variant (Boolean)

Syntax

value **=** *range*.**IsRowCollapsed**

Legal values

Value	Description
TRUE	Rows in the range are collapsed.
FALSE	Rows in the range are not collapsed.

1-2-3: IsRowHidden property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ISCOLUMNHIDDEN_AND_ISROWHIDDEN_PROPERTIES_EXSCRIPT ',1)} See example

(Read-only) Determines whether rows in a specified range are hidden.

Data type

Variant (Boolean)

Syntax

value **=** *range*.**IsRowHidden**

Legal values

Value	Description
TRUE	Rows in the range are hidden.
FALSE	Rows in the range are displayed.

1-2-3: IsSelectable property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_ARC_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

(Read-only) Determines whether an object can be added to a selection.

Data type

Variant (Boolean)

Syntax

value **=** *object*.**IsSelectable**

Legal values

Value	Description
TRUE	The object can be added to a selection.
FALSE	The object cannot be added to a selection.

1-2-3: IsSelected property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS;H_123_QUERY_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

(Read-only) Determines whether an object is selected.

Data type

Variant (Boolean)

Syntax

value **=** *object*.**IsSelected**

Legal values

Value	Description
TRUE	Object is selected.
FALSE	Object is not selected.

1-2-3: IsZeroDisplayed property

{button ,AL(`;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ISZERODISPLAYED_PROPERTY_EXSCRIPT;H_123_DISPLAYZEROAS_PROPERTY_EXSCRIPT;',1)} See example

Determines whether zeros are displayed in the sheet.

Data type

Variant (Boolean)

Syntax

sheet.**IsZeroDisplayed =** *value*

value **=** *sheet*.**IsZeroDisplayed**

Legal values

Value	Description
TRUE	(Default) Display zeros in the sheet.
FALSE	Do not display zeros in the sheet.

```
' Example: DisplayZeroAs, IsZeroDisplayed, and Sheets properties
' Display all zeros in a workbook as the word "None."
Sub ChangeZeros
   'Act on all sheets in the current workbook
   ForAll x in CurrentDocument.Sheets
      'Turn on the display of zeros and display all zeros as the word "None."
      x.IsZeroDisplayed = True
      x.DisplayZeroAs = "None"
   End ForAll
End Sub
```

1-2-3: Italic property

{button ,AL(`;H_123_FONT_CLASS',0)} See list of classes

{button ,AL(`H_123_ITALIC_PROPERTY_EXSCRIPT ',1)} See example

Determines whether data is styled using italics.

Data type

Variant (Boolean)

Syntax

font.**Italic =** *value*

value **=** *font*.**Italic**

Legal values

Value	Description
TRUE	Apply italics.
FALSE	Do not apply italics.

```
' Example: ColorName, DoubleUnderline, Font, FontColor, FontName, Italic and Size
properties
' Formats the selected range. This format would be appropriate for a title.
Sub TitleStyle
   Selection.Font.FontName = "Century Schoolbook"
   Selection.Font.FontColor.ColorName = "50% gray"
   Selection.Font.Italic = True
   Selection.Font.DoubleUnderline = True
   Selection.Font.Size = 18
End Sub
```

1-2-3: Keywords property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_KEYWORDS_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns keywords associated with a file.

Data type

String

Syntax

document.**Keywords =** *value*

value **=** *document*.**Keywords**

Legal values

The value of the Keywords property is a string containing the keywords you set or entered using File - Workbook Properties (General tab).

```
' Example: Author, Contents, Keywords, and LastEditor properties; MoveCellPointer
method
' Enter 1-2-3 Statistics information in a sheet.
Sub Stats
   ' Assign variables for the current file's keywords,
   ' author, and last editor.
   kw = CurrentDocument.Keywords
   au = CurrentDocument.Author
   le = CurrentDocument.LastEditor
   ' Enter the file's keywords, author, and last editor in
   ' separate cells down a column.
   Selection.Contents = kw
   [A].MoveCellPointer $Down,1
   Selection.Contents = au
   [A].MoveCellPointer $Down,1
   Selection.Contents = le
End Sub
```

1-2-3: KnownRegionAliases property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS',0)} See list of classes

(Read-only) Returns a collection of available custom names for recognized map regions for a map. For example, you may have created the custom name "Nippon" for the recognized region name "Japan" for the World Countries map.

Data type

Strings

Syntax

Set *strings* **=** *object*.**KnownRegionAliases**

Legal values

The value of the KnownRegionAliases property is a collection of strings containing custom names you created for regions of the map.

1-2-3: KnownRegionCodes property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS',0)} See list of classes

(Read-only) Returns a collection of available map region codes for a map.

Data type

Strings

Syntax

Set *strings* **=** *object*.**KnownRegionCodes**

Legal values

The value for the KnownRegionCodes property is a collection of strings containing the map region codes for the map.

1-2-3: KnownRegionNames property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS',0)} See list of classes

(Read-only) Returns a collection of available region names for a map.

Data type

Strings

Syntax

Set *strings* **=** *object*.**KnownRegionNames**

Legal values

The value for the KnownRegionNames property is a collection of strings containing names for the regions of the map.

1-2-3: L123Seconds property

{button ,AL(`H_123_DATETIME_CLASS;',0)} See list of classes

(Read-only) Returns the internal 1-2-3 representation of the date and time specified by the DateTime object.

Data type

Double

Syntax

value **=** *datetime*.**L123Seconds**

Legal values

The L123Seconds property returns a 1-2-3 time number. The integer portion of this positive number represents a date; the fractional portion represents a time.

Usage

The L123Seconds property is used by 1-2-3 to communicate with the DateTime object.

1-2-3: LabelRange property

{button ,AL(`H_123_MAPBINS_CLASS',0)} See list of classes

Sets or returns the range that contains the legend labels for the specified map data bin.

Data type

Range

Syntax

Set *mapbins*.**LabelRange =** *range*

Set *range* **=** *mapbins*.**LabelRange**

Legal values

The value of the LabelRange property is a Range object containing labels.

Usage

The LabelRange property is ignored unless the LabelSource property is set to $LabelsFromRange. Because there is a maximum of six map bins, only the first six values in the range are used.

1-2-3: LabelSource property

{button ,AL(`H_123_MAPBINS_CLASS',0)} See list of classes

(Read-only) Returns the way in which legend labels are defined.

Data type

Variant (MapBinLabelSource enumeration)

Syntax

value **=** *mapbins*.**LabelSource**

Legal values

Value	Description
$FromValues	1-2-3 determines the legend labels from the bin values.
$LabelsManual	You specify the label strings.
$LabelsFromRange	You specify the range that contains the legend labels.

1-2-3: LastEditor property

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;',0)} See list of classes

{button ,AL(`H_123_KEYWORDS_PROPERTY_EXSCRIPT;H_123_EDITINGTIME_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the name of the user who made the last change to the file, version, or version group.

Data type

String

Syntax

value **=** *object*.**LastEditor**

Legal values

The default value for the LastEditor property is the name of the user who made the last change to the file, version, or version group. If no revision has been made, the value is the same as the value of the Author property.

1-2-3: LastPrinted property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_LASTPRINTED_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns the system date and time when the file was last printed.

Data type

DateTime

Syntax

Set *datetime* **=** *document*.**LastPrinted**

Legal values

The value of the LastPrinted property is the date and time on your system when the file was last printed.

```
' Example: LastPrinted property
' Prints the current workbook if it was not printed the last time it was saved, using
the current print and page settings.
Sub PrintDoc
   If CurrentDocument.LastPrinted.LocalTime <> _
      CurrentDocument.LastPrinted.LocalTime Then
      CurrentApplication.Print
End If
End Sub
```

1-2-3: LastVersionGroup property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

(Read-only) Returns the last version group that was current in the file.

Data type

VersionGroup

Syntax

Set *versiongroup* **=** *document*.**LastVersionGroup**

Legal values

The value of the LastVersionGroup property is the last version group you set or the last version group you displayed using Range - Version Group.

1-2-3: LeftBorder property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_LEFTBORDER_PROPERTY_EXSCRIPT ',1)} See example

With the Style or Color property, sets or returns the style or color of the left border of a range.

Data type

RangeBorder

Syntax

Set *range*.**LeftBorder =** *rangeborder*

range.**LeftBorder.Style =** *value*

Set *range*.**LeftBorder.Color =** *value*

Set *rangeborder* **=** *range*.**LeftBorder**

value **=** *range*.**LeftBorder.Style**

Set *value* **=** *range*.**LeftBorder.Color**

Legal values

A RangeBorder object.

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

```
' Example: Color, ColorName, and LeftBorder properties
' Adds a light lavender, double border to the left of a selected range.
Sub StyleLeftBorder
   Selection.LeftBorder.Style = $DoubleBorder
   Selection.LeftBorder.Color.ColorName = "light lavender"
End Sub
```

1-2-3: LeftMargin property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the left margin print setting, in twips.

Data type

Long

Syntax

printsettings.**LeftMargin =** *value*

value **=** *print*settings.**LeftMargin**

Legal values

The value of the LeftMargin property is the left margin you set or the left margin you specified using File - Preview & Setup (Layout tab).

1-2-3: Left property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_LEFT_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the coordinate of the left boundary of the bounding rectangle for a graphic object, in twips, or the horizontal coordinate of the window's origin.

Data type

Long

Syntax

object.**Left =** *value*

value **=** *object*.**Left**

Legal values

The boundary for a graphic object cannot be outside the boundary for the worksheet.

Usage

If the window is minimized or maximized, you will not see the effect of the Left property setting until the window is restored.

```
' Example: Left property
' Align the left edges of all the charts on a sheet to make the
' data look neater when printed.
Sub LineUp
   'This script acts on all charts in the current workbook.
   ForAll x in CurrentDocument.Charts
      'Place the left edge of all charts 1000 pixels from the left edge of the window.
      x.Left = 1000
   End ForAll
End Sub
```

1-2-3: Legend property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS',0)} See list of classes

(Read-only) Returns the legend object for the specified map.

Data type

Legend

Syntax

Set *legend* **=** *object*.**Legend**

Legal values

The return value for the Legend property is a legend object.

1-2-3: LinkedToCell property

{button ,AL(`H_123_MAPTEXTENTRY_CLASS',0)} See list of classes

Indicates whether a specifed line of text is connected to a cell.

Data type

Variant (Boolean)

Syntax

maptextentry.**LinkedToCell** = *value*

Legal values

Value	Description
TRUE	The text is connected to a cell.
FALSE	The text is not connected to a cell.

1-2-3: LinkSource property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

Sets or returns the source of a link.

Data type

String

Syntax

object.**LinkSource =** *value*

value **=** *object*.**LinkSource**

Legal values

The default value for the LinkSource property is the source you set or the source of a link you specified using the SetLinkSource method.

1-2-3: LocalTime property

{button ,AL(`;H_123_DATETIME_CLASS',0)} See list of classes

Sets or returns the time on your computer as a string.

Data type

String

Syntax

datetime.**LocalTime =** *value*

value **=** *datetime*.**LocalTime**

Legal values

The value for the LocalTime property is the time you set or the time you specified in the Windows Control Panel. You can set time in hours, minutes, and seconds.

1-2-3: Location property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

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

(Read-only) Returns the location of the file, either a directory or a Notes database. 1-2-3 copies the location into a temp file.

Data type

String

Syntax

value **=** *document*.**Location**

Legal values

The value for the Location property is the path in which the file is located.

```
' Example: Location property
' Finds a file to create an automatic backup.
```

1-2-3: LongPrompt property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Sets or returns the current long prompt. A long prompt is a description that appears in the title bar of the 1-2-3 window when you highlight a commmand.

Data type

String

Syntax

applicationwindow.**LongPrompt =** *value*

value **=** *applicationwindow*.**LongPrompt**

Legal values

The value of the LongPrompt property is a string containing the text of the prompt.

1-2-3: LSGMTTime property

{button ,AL(`H_123_DATETIME_CLASS',0)} See list of classes

(Read-only) Returns the LotusScript date value in Greenwich mean time.

Data type

Variant

Syntax

value **=** *datetime*.**LSGMTTime**

Legal values

The value for the LSGMTTime property is any valid date.

1-2-3: LSLocalTime property

{button ,AL(`H_123_DATETIME_CLASS',0)} See list of classes

Sets or returns the LotusScript date value for the current date on your system.

Data type

Variant

Syntax

datetime.**LSLocalTime =** *value*

value **=** *datetime*.**LSLocalTime**

Legal values

The value of the LSLocalTime property is the time you set or the time you specified in the Windows Control Panel.

1-2-3: MacroStep property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether to step through each instruction of a macro one at a time.

Data type

Variant (Boolean)

Syntax

application.**MacroStep =** *value*

value **=** *application*.**MacroStep**

Legal values

Value	Description
TRUE	Step through each instruction of the macro.
FALSE	(Default) Do not step through each instruction of the macro.

Usage

Setting the value of the MacroStep property to TRUE automatically sets the value of the MacroTrace property to TRUE.

1-2-3: MacroTrace property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether to display the Macro Trace window to view a macro while it is running.

Data type

Variant (Boolean)

Syntax

application.**MacroTrace =** *value*

value **=** *application*.**MacroTrace**

Legal values

Value	Description
TRUE	Display the Macro Trace window.
FALSE	(Default) Do not display the Macro Trace window.

1-2-3: MaintainDimensions property

{button ,AL(`H_123_PLOT_CLASS;',0)} See list of classes

Determines whether to maintain the correct dimensions of the map when you resize it.

Data type

Variant (Boolean)

Syntax

mapplot.**MaintainDimensions =** *value*

value **=** *mapplot*.**MaintainDimensions**

Legal values

Value	Description
TRUE	(Default) Maintain the correct map dimensions.
FALSE	Do not maintain the correct map dimensions.

1-2-3: Maps property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_MAPS_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns a collection of the maps for the specified file.

Data type

Maps

Syntax

Set *maps* **=** *document*.**Maps**

Legal values

The value of the Maps property is the maps collection for the file.

```
' Example: Maps property
' Clears all maps.
 Forall mps in CurrentDocument.Maps
    mps.Clear
 End Forall
```

1-2-3: MatchAccent property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_MATCHCASE_PROPERTY_MATCHACCENT_PROPERTY_EXSCRIPT;',1)} See example

Determines whether to look for accented characters when finding text.

Data type

Variant (Boolean)

Syntax

application.**MatchAccent =** *value*

value **=** *application*.**MatchAccent**

Legal values

Value	Description
TRUE	Look for accented characters.
FALSE	(Default) Do not look for accented characters.

1-2-3: MatchCase property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_MATCHCASE_PROPERTY_MATCHACCENT_PROPERTY_EXSCRIPT;',1)} See example

Determines whether to look for characters using the exact combination of upercase and lowercase letters you enter when finding text.

Data type

Variant (Boolean)

Syntax

application.**MatchCase =** *value*

value **=** *application*.**MatchCase**

Legal values

Value	Description
TRUE	Look for exact combination of uppercase and lowercase letters.
FALSE	(Default) Do not look for exact combination of uppercase and lowercase letters.

```
'Example: MatchAccent and MatchCase properties
' Open a new document and call it TestDocument.
Dim TestDocument As Document
Set TestDocument = CurrentApplication.NewDocument("TestDocument")
' Add some data to cells A:A1 through A:A5 in TestDocument.
[A:A1].Select
Selection.Contents = "Test document for example"
[A].MoveCellPointer $Down,1
Selection.Contents = "crepe paper"
[A].MoveCellPointer $Down,1
Selection.Contents = "crêpe desserts"
[A].MoveCellPointer $Down,1
Selection.Contents = "Crepe de Chine"
' Specify search and replace strings
CurrentApplication.SearchString = "crepe"
CurrentApplication.ReplaceString = "green"
' Specify search characteristics
CurrentApplication.MatchAccent = True
CurrentApplication.MatchCase = False
' Replace the first occurence
Messagebox("Replace ""crepe"" with ""green"" in the first occurence.")
[A1..A5].Replace
Messagebox("Replace ""crepe"" with ""green"" in all occurences.")
[A1..A5].ReplaceAll
```

1-2-3: MatchKatakana property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether specified text is Katakana-to-Hiragana sensitive.

Data type

Variant (Boolean)

Syntax

application.**MatchKatakana =** *value*

value **=** *application*.**MatchKatakana**

Legal values

Value	Description
TRUE	The specified text is Katakana-to-Hiragana sensitive.
FALSE	(Default) The specified text is not Katakana-to-Hiragana sensitive.

1-2-3: MatchPitch property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether to check if pitch matches when finding text.

Data type

Variant (Boolean)

Syntax

application.**MatchPitch =** *value*

value **=** *application*.**MatchPitch**

Legal values

Value	Description
TRUE	The specified text is pitch-sensitive.
FALSE	(Default) The specified text is not pitch-sensitive.

1-2-3: MenuPrompt property

{button ,AL(`H_123_MENU_CLASS;H_123_MENUBAR_CLASS;',0)} See list of classes

Sets or returns the long prompt for a selected menu item. This property is read-write for menu commands you create with LotusScript, and read-only for built-in 1-2-3 menus.

Data type

String

Syntax

object.**MenuPrompt =** *value*

value **=** *object*.**MenuPrompt**

Legal values

The value of the MenuPrompt property is a string containing the text of the prompt.

1-2-3: Methods property

{button ,AL(`H_123_CLASSINFO_CLASS',0)} See list of classes

(Read-only) Returns a collection of the available methods for the class.

Data type

Strings

Syntax

Set *strings* **=** *classinfo*.**Methods**

Legal values

The value of the Methods property is a collection of strings containing the names of the methods for the class.

1-2-3: ModifiedDate property

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS',0)} See list of classes

{button ,AL(`H_123_MODIFIEDDATE_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the most recent date the file, version, or version group was modified.

Data type

DateTime

Syntax

Set *datetime* **=** *object*.**ModifiedDate**

Legal values

If the file, version, or version group has not been modified, the default value is the date it was created.

```
' Example: ModifiedDate property
' Check if file has been modified.
Dim File1 As Document
Dim ModifiedDateResult As String
Set File1 = CurrentApplication.NewDocument
File1.SaveAs "test123"
If File1.CreationDate.LocalTime = File1.ModifiedDate.LocalTime Then
ModifiedDateResult = "Not Modified"
Else
ModifiedDateResult = "Modified"
End If
```

1-2-3: MonthNames property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_MONTHNAMES_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the unabbreviated names of the months used in dates.

Data type

Strings

Syntax

Set *strings* **=** *application*.**MonthNames**

Legal values

The default values of the MonthNames property are the names of the months determined by the country setting you specified in the Windows Control Panel.

```
' Example: Contents and MonthNames properties
Forall x In CurrentApplication.MonthNames
Selection.Contents = x
.MoveCellPointer $Down, 1
End Forall
```

1-2-3: MorningString property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns a time string for the AM symbol.

Data type

String

Syntax

value **=** *application*.**MorningString**

Legal values

The default value of the MorningString property is AM, or a string you specified in the Windows Control Panel.

1-2-3: NamedPrintSettings property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_NAMEDPRINTSETTINGS_PROPERTY_EXSCRIPT',1)} See example

Sets or returns a collection of the named print styles.

Data type

PrintSettingsCollection

Syntax

Set *document*.**NamedPrintSettings =** *printsettingscollection*

Set *printsettingscollection* **=** *document*.**NamedPrintSettings**

Legal values

The default value for the NamedPrintSettings property is a collection of the names of the print styles you set, or the names of the print styles you created using File - Preview & Page Settings (Named Styles).

A document always has a PrintSettings object named Default.

```
' Example: NamedPrintSettings property
' Print a list of named print styles in the current file.
Dim printcollect As PrintSettingsCollection
Set printcollect = CurrentDocument.NamedPrintSettings
Forall PSC In printcollect
Print PSC.Name
End Forall
```

1-2-3: NamedRanges property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_NAMEDRANGES_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns a list of all named ranges in the file.

Data type

Strings

Syntax

Set *strings* **=** *document***.NamedRanges**

Legal values

The value of the NamedRanges property is a collection of the range name strings associated with the file.

```
' Example: NamedRanges property
' Print in the Output view all of the named ranges in the workbook.
Forall ranges In CurrentDocument.NamedRanges
Print ranges
End Forall
```

1-2-3: Name property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATAQUERY_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_NAME_PROPERTY_EXSCRIPT;H_123_SHEETCOUNT_PROPERTY_EXSCRIPT;H_123_SELECTION_PROPERTY_EXSCRIPT;H_123_SHEETS_PROPERTY_EXSCRIPT;H_123_GREEN_PROPERTY_EXSCRIPT;H_123_ISRANGENAMED_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the name of the object.

Data type

String

Syntax

value **=** *object***.Name**

Legal values

This property is read-write for certain classes, such as the Query class.

For the Range class, the default value for the Name property when a range has multiple names associated with it is the last range name used. If a range address was used, the address will be returned.

For the Document class, the default value for the Name property is the name of the file, including the path.

If the object is not named, the Name property returns the name of the parent object.

For collections, the Name property returns an empty string.

```
' Example: Name property
' Display the name of the current document in a message box.
Dim file1 As Document
Set file1 = CurrentApplication.NewDocument
Dim filename As String
filename = CurrentDocument.Name
MessageBox filename
```

1-2-3: NegativesInColor property

{button ,AL(`H_123_RANGE_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_NEGATIVESINCOLOR_PROPERTY_EXSCRIPT;',1)} See example

Determines whether negative numbers are displayed in red in the range or sheet.

Data type

Variant (Boolean)

Syntax

object.**NegativesInColor =** *value*

value **=** *object*.**NegativesInColor**

Legal values

Value	Description
TRUE	Display negative numbers in red.
FALSE	(Default) Do not display negative numbers in red.

```
' Example: ColorIndex, FormatDecimals, FormatName, NegativesInColor, and
' OutlineBorder properties; RangeFill method
' Set up and format a range, using various styles and number formats.
Dim testrange As Range
Dim testborder As RangeBorder
Set testrange = Bind("A:A1..A:C20")
testrange.RangeFill -5,1,32767,$Number,
Set testborder = testrange.OutlineBorder
testborder.Style = $ThickBorder
testborder.Color.ColorIndex = 54
testrange.NegativesInColor = True
testrange.Font.Bold = True
testrange.FormatDecimals = 2
testrange.FormatName = "US Dollar"
MsgBox "Range has been filled and formatted."
```

1-2-3: Normal property

{button ,AL(`H_123_FONT_CLASS',0)} See list of classes

Sets or returns whether the data is styled using the normal attribute.

Data type

Variant (Boolean)

Syntax

font.**Normal** = *value*

value = *font*.**Normal**

Legal values

Value	Description
TRUE	Apply the normal attribute to the data. This sets all other font attributes to FALSE.
FALSE	Do not apply the normal attribute to the data. You cannot explicitly set this value; FALSE is returned when another text property (for example, Bold) is set to TRUE.

1-2-3: NotesPath property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_NOTESPATH_PROPERTY_EXSCRIPT',1)} See example

(Read only) Returns the path where Notes is installed.

Data type

String

Syntax

filename **=** *application***.Notespath**

Legal values

The value for the NotesPath property is a string containing a path name and executable filename, for example, "C:\NOTES\NOTES.EXE".

```
' Example: NotesPath property
' Displays a message box showing the location of Notes.

Dim x as String
x = CurrentApplication.NotesPath
MessageBox (x)
```

1-2-3: NumberOfMostRecentFiles property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_NUMBEROFMOSTRECENTFILES_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the number of the most recently opened files to display on the File menu.

Data type

Long

Syntax

application.**NumberOfMostRecentFiles =** *value*

value **=** *application*.**NumberOfMostRecentFiles**

Legal values

The value of the NumberOfMostRecentFiles property can be any Long integer from 0 - 10.

```
' Example: NumberOfMostRecentFiles property
' Returns the number of files listed.
Dim myapp As Application
Dim filenumber As Integer
Set myapp = CurrentApplication
' Store the default number of most recent files
filenumber = myapp.NumberOfMostRecentFiles
Messagebox ("The number of most recent files that will be " + _
 "displayed on the File menu is: " + Str(filenumber))
```

1-2-3: Object property

{button ,AL(`H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;H_123_QUERYTABLE_CLASS;H_123_APPROACHCONNECTION_CLASS',0)} See list of classes

{button ,AL(`h_123_object_property_exscript',1)} See example

(Read-only) Returns a reference to the OLE Automation object.

Data type

Variant

Syntax

Set *object* **=** *objectspecifier***.Object**

Legal values

A reference to the object specified by *objectspecifier*.

Usage

Use the Object property to call the automation commands and properties supported by the OLE Automation class. Use the Set statement to assign the object reference returned to a Variant variable. This is the same type returned by the functions CreateObject and GetObject.

```
' Example: Object property
' Use the Object property to get an OLE automation server object.

' Example 1.  Word Pro automation server
' Embed a new Word Pro document using OLE automation, add content, and save it.
' Note:  To run this example, OLE automation needs to be unchecked
'        in the Disable drop-down list in Word Pro Preferences.
Dim myObj As OLEObject
Dim myDoc As Variant
' First, create a Word Pro OLE object, of 1-2-3 type OLEObject.
Set myObj = [A].NewObject(1080, 360, 6480, 4320, "WordPro.Document",,, False,,,)
' Get the Word Pro TextDocument object for OLE automation.
Set myDoc = [OLE 1].Object
' Display the object's content.
myObj.Verb $OLEVerbShow
' Now run methods of the TextDocument class in Word Pro.
' For example, use a different SmartMaster.
myDoc.Parent.ChangeSmartMaster "f:\lotus\smasters\wordpro\letter1.mwp", "", "Current
division only"
' Add some text.
myDoc.Parent.Type "This is a test of using Word Pro as an OLE automation server. "
' Save the Word Pro document to a file.
myDoc.SaveAs "F:\lotus\work\wordpro\123ObjP.lwp", "", "Lotus Word Pro", False, True,
False

' Example 2.  Freelance Graphics automation server
' Embed a Freelance Graphics document and add some content to it.
' First, create the Freelance Graphics OLE object, of 1-2-3 type OLEObject.
[A].NewObject 1080, 720, 6480, 5080, "FLW3Presentation",,, False,,,
' At this point, Freelance Graphics is launched full-screen,
' and presents two modal dialog boxes asking the user to select
' a SmartMaster Content topic and Look, and a page layout for
' the new presentation.
' Get the Freelance Graphics Document object for OLE automation.
Dim mydoc As Variant       ' Freelance Graphics Document object
Set mydoc = [OLE 1].Object
' Now run methods and set properties of the Document class in Freelance Graphics.
' For example:
' Set the SmartMaster look.
mydoc.SmartLook = "blank.mas"
' Create a new page.
mydoc.CreatePage "my page", 12
' Add a block of text.
Dim txtblock As Variant
Set txtblock = mydoc.Pages("my page").CreateText(2880, 5760, 7200, 3600)
txtblock.Text = "Page content."
```

1-2-3: OLEObjects property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`h_123_oleobjects_property_exscript',1)} See example

(Read-only) Returns a list of the OLE objects embedded in the specified file.

Data type

OLEObjects

Syntax

Set *oleobjects* **=** *document***.OLEObjects**

Legal values

The value of the OLEObjects property is a collection of references to the OLE objects in the file.

```
' Example: OLEObjects property
' Check all of the OLE objects and, for linked OLE objects, update the link.
Forall foo In CurrentDocument.OLEObjects
If foo.Islinked = True Then
foo.Update
End If
End Forall
```

1-2-3: Orientation property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the orientation of paper for printed output.

Data type

Variant (Orientation enumeration)

Syntax

printsettings.**Orientation =** *value*

value **=** *printsettings*.**Orientation**

Legal values

Value	Description
$Portrait	(Default) Print output in the portrait (vertical) orientation.
$Landscape	Print output in the landscape (horizontal) orientation.

1-2-3: OSType property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the name and version of the operating system installed on your computer.

Data type

String

Syntax

value **=** *application*.**OSType**

Legal values

The value of the OSType property is a string containing the name and version of the operating system installed on your computer.

1-2-3: OutlineBorder property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_OUTLINEBORDER_PROPERTY_EXSCRIPT;H_123_NEGATIVESINCOLOR_PROPERTY_EXSCRIPT',1)} See example

With the Style or Color property, sets or returns the style or color of the outline border of a range.

Data type

RangeBorder

Syntax

Set *range*.**OutlineBorder =** *rangeborder*

Set *range*.**OutlineBorder.Style =** *value*

Set *range*.**OutlineBorder.Color =** *color*

Set *rangeborder* **=** *range*.**OutlineBorder**

Set *value* **=** *range*.**OutlineBorder.Style**

Set *color* **=** *range*.**OutlineBorder.Color**

Legal values

A RangeBorder object.

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

```
' Example: OutlineBorder property and RangeFill method
' Set up range, fill it with numbers, and give it a solid outline border.
Dim testrange As Range
Dim testborder As RangeBorder
Set testrange = Bind("A:A1..A:C20")
testrange.RangeFill 0,1,32767,$Number,
Set testborder = testrange.OutlineBorder
testborder.Style = $ThickBorder
```

1-2-3: OutputLocation property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Sets or returns the address of the output range for the query table.

Data type

String

Syntax

dataquery.**OutputLocation** = *value*

value = *dataquery*.**OutputLocation**

Legal values

The value of the OutputLocation property is a string containing the range address of the output range for the specified query table.

1-2-3: OutputRange property

{button ,AL(`H_123_QUERYTABLE_CLASS',0)} See list of classes

Sets or returns the output range for a query table.

Data type

Range

Syntax

Set *querytable*.**OutputRange =** *range*

Set *range* **=** *querytable*.**OutputRange**

Legal values

The value for the OutputRange property is a valid range.

1-2-3: Overlays property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS;',0)} See list of classes

(Read-only) Returns the names of the overlays that are added to a base map.

Data type

Strings

Syntax

Set *strings* **=** *object***.Overlays**

Legal values

The values for the Overlays property are the names of the overlays you set or the names of the overlays you specified using Map - Map Properties (Overlays).

1-2-3: PaperBinNames property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns a collection of all the printer paper sources supported by the current printer.

Data type

Strings

Syntax

Set *printsettings*.**PaperBinNames =** *strings*

Set *strings* **=** *printsettings*.**PaperBinNames**

Legal values

The bin names in the PaperBinNames collection can include the following:

AutoSelect

UpperTray

ManualFeed

EnvelopeManualFeed

LowerTray

Envelope

1-2-3: PaperBinName property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the name of a printer paper source.

Data type

String

Syntax

printsettings.**PaperBinName =** *value*

value **=** *printsettings*.**PaperBinName**

Legal values

The default values for the PaperBinName property are the following:

AutoSelect

UpperTray

ManualFeed

EnvelopeManualFeed

LowerTray

Envelope

1-2-3: PaperHeightMaximum property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

(Read-only) Returns the maximum height of the printer paper, in twips. This is a read-write property if the paper size has been set to a custom size.

Data type

Long

Syntax

printsettings.**PaperHeightMaximum =** *value* (for custom size)

value **=** *printsettings*.**PaperHeightMaximum**

Legal values

The default value for the PaperHeightMaximum property is the height of the paper size you specified using File - Preview & Page Setup (Printer).

1-2-3: PaperHeightMinimum property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

(Read-only) Returns the minimum height of the printer paper, in twips. This is a read-write property if the paper size has been set to a custom size.

Data type

Long

Syntax

printsettings.**PaperHeightMinimum =** *value* (for custom size)

value **=** *printsettings*.**PaperHeightMinimum**

Legal values

The default value for the PaperHeightMinimum property is the height of the paper size you specified using File - Preview & Page Setup (Printer).

1-2-3: PaperHeight property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

(Read-only) Returns the height of the printer paper, in twips. This is a read-write property if the paper size has been set to a custom size.

Data type

Long

Syntax

printsettings.**PaperHeight =** *value* (for custom size)

value **=** *printsettings*.**PaperHeight**

Legal values

The default value for the PaperHeight property is what you set or what you specified using File - Preview & Page Setup (Printer).

1-2-3: PaperSizeNames property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

(Read-only) Returns a collection of strings for all the paper sizes supported by the current printer.

Data type

Strings

Syntax

value **=** *printsettings*.**PaperSizeNames**

Legal values

The value for the PaperSizeNames property is the collection of paper size names supported by the current printer.

1-2-3: PaperSizeName property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the name of a paper size.

Data type

String

Syntax

printsettings.**PaperSizeName =** *value*

value **=** *printsettings*.**PaperSizeName**

Legal values

The following table shows the names for some common paper sizes.

Value	Description
Letter	(Default) 8 1/2 x 11 inches
Legal	8 1/2 x 14 inches
Executive	7 1/4 x 10 1/2 inches
A4	219 x 297 millimeters
Envelope # 10	4 1/8 x 9 1/2 inches
Envelope DL	110 x 220 millimeters
Envelope Monarch	3 7/8 x 7 1/2 inches

1-2-3: PaperWidthMaximum property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

(Read-only) Returns the maximum width of the printer paper, in twips. This is a read-write property if the paper size has been set to a custom size.

Data type

Long

Syntax

printsettings.**PaperWidthMaximum =** *value* (for custom size)

value **=** *printsettings*.**PaperWidthMaximum**

Legal values

The default value for the PaperWidthMaximum property is the width of the paper size you specified using File - Preview & Page Setup (Printer).

1-2-3: PaperWidthMinimum property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

(Read-only) Returns the minimum width of the printer paper, in twips. This is a read-write property if the paper size has been set to a custom size.

Data type

Long

Syntax

printsettings.**PaperWidthMinimum =** *value* (for custom size)

value **=** *printsettings*.**PaperWidthMinimum**

Legal values

The default value for the PaperWidthMinimum property is the width of the paper size you specified using File - Preview & Page Setup (Printer).

1-2-3: PaperWidth property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

(Read-only) Returns the width of the printer paper, in twips. This is a read-write property if the paper size has been set to a custom size.

Data type

Long

Syntax

printsettings.**PaperWidth =** *value* (for custom size)

value **=** *printsettings*.**PaperWidth**

Legal values

The default value for the PaperWidth property is the width of the paper size you specified using File - Preview & Page Setup (Printer).

1-2-3: Parent property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATAQUERY_CLASS;H_123_DATETIME_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASSH_123_MAPPLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

{button ,AL(`H_123_PARENT_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the parent object of the current or specified object.

Data type

Variant

Syntax

value **=** *object*.**Parent**

Legal values

The value for the Parent property is the containing object. For example, the value for the Parent property of a Document object is the Application object.

```
' Example: Name and Parent properties
' Shows what sheet the first chart is on.
If CurrentDocument.Charts.Count > 0 Then
   Print CurrentDocument.Charts(0).Parent.Name
Else
   Print "No Charts"
End If
```

1-2-3: Password property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_PASSWORD_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the password needed to open the current file.

Data type

String

Syntax

document.**Password =** *value*

value **=** *document*.**Password**

Legal values

The value for the Password property is any string of 15 or fewer characters. The value for the Password property is what you set or the password you specified using File - Save As.

```
' Example: Password property; Save and SaveAs methods
' Set a password.
Dim mydoc As Document
Dim mypasswd As String
Set mydoc = CurrentApplication.NewDocument()
mydoc.SaveAs "FileOne"
mypasswd = "itsasecret"
'Set the password and then save the file.
mydoc.Password = mypasswd
mydoc.Save
```

1-2-3: Path property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_PATH_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the full path of a file or application.

Data type

String

Syntax

value **=** *document*.**Path**

or

value **=** *application*.**Path**

Legal values

The default value for the Path property is the location of the file or application, including the full path.

```
' Example: Path property
' Find path and print.
Dim  mydoc As Document
Dim  mypath As String
Set  mydoc = CurrentDocument
my path = mydoc.Path
Print  mypath
```

1-2-3: PatternBins property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS;',0)} See list of classes

(Read-only) Returns a collection of the pattern bins for the specified map.

Data type

MapBins

Syntax

value = *object*.**PatternBins**

Legal values

The default value for the PattenBins property is a MapBins object containing the pattern bins for the specified map.

1-2-3: PatternVisible property

{button ,AL(`H_123_LEGEND_CLASS',0)} See list of classes

Determines whether to display the specified pattern legend for a map.

Data type

Variant (Boolean)

Syntax

legend.**PatternVisible =** *value*

value **=** *legend*.**PatternVisible**

Legal values

Value	Description
TRUE	(Default) Display the pattern legend.
FALSE	Do not display the pattern legend.

1-2-3: Pattern property

{button ,AL(`H_123_BACKGROUND_CLASS;H_123_MAPBIN_CLASS',0)} See list of classes

{button ,AL(`H_123_PATTERN_PROPERTY_EXSCRIPT;H_123_BACKCOLOR_PROPERTY_EXSCRIPT;H_123_SHOWDRAWLAYER_PROPERTY_EXSCRIPT;H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the fill region pattern for Background and MapBin objects.

Data type

Variant (PatternType enumeration)

Syntax

object.**Pattern =** *value*

value **=** *object*.**Pattern**

Legal values

Value	Description
$BowlingBalls	
$BroadRightHatch	
$Checkers	
$CoarseCrossHatch	
$CoarseSquareHatch	
$Confetti	
$CrossHatch	
$DarkCoarseCrossHatch	
$DarkSingleLeftHatch	
$DarkSingleRightHatch	
$DiagonalBricks	
$Diamonds	
$DottedSquareHatch	
$DoubleCoarseLeftHatch	
$DoubleCoarseRightHatch	
$DoubleHorizStripes	
$DoubleLeftHatch	
$DoubleRightHatch	
$DoubleVertStripes	
$FineCheckers	
$FineCrossHatch	
$FineSquareHatch	
$Flags	
$Fountains	
$GradientFillHorizBG2FG	shades horizontally from light to dark
$GradientFillHorizFG2BG	shades horizontally from dark to light
$GradientFillVertBG2FG	shades vertically from dark to light
$GradientFillVertFG2BG	shades vertically from light to dark
$GrayScale2ndDarkest	
$GrayScale2ndLightest	

$GrayScale3rdDarkest 
$GrayScale3rdLightest 
$GrayScale4rdDarkest 
$GrayScale4thLightest 
$GrayScale5thDarkest 
$GrayScale5thLightest 
$GrayScale6thDarkest 
$GrayScaleDarkest 
$GrayScaleLightest 
$HorizCoarseHatch 
$HorizHatch 
$LongHorizCheckers 
$Mountains 
$Pavers 
$QuadrupleDarkHorizStripes 
$QuadrupleDarkVertStripes 
$QuadrupleGrayHorizStripes 
$QuadrupleGrayVertStripes 
$RoundStones 
$RunningBricks 
$SingleCoarseLeftHatch 
$SingleCoarseRightHatch 
$SingleLeftHatch 
$SingleNarrowLeftHatch 
$SingleRightHatch 
$SolidBackground  (background color)
$SolidForeground  (foreground color)
$SquareHatch 
$TallVertCheckers 
$Transparent 
$TripleLeftHatch 
$TripleRightHatch 
$VertCoarseHatch 
$VertHatch 
$Waves 
$WideCrossHatch 
$WideSquareHatch 
$Zigzags 

```
' Example: Background, ColorName, ColumnWidth, Font, FontColor, Pattern, and
' RowHeight properties; RangeFill and Select methods
' Select a range, adjust its style, add numbers, add a background pattern,
' and change font color.
Dim rng1 as Range
Set rng1 = [A:B5..A:H25]
rng1.RowHeight = 20
rng1.ColumnWidth = 15
rng1.RangeFill 200,3,37999,$Number
rng1.Background.Pattern = $DoubleRightHatch
rng1.Background.BackColor.ColorName = "pale green"
rng1.Font.FontColor.ColorName = "black"
rng1.Font.Bold = True
```

1-2-3: Placement property

{button ,AL(`H_123_MAPTITLE_CLASS;H_123_LEGEND_CLASS',0)} See list of classes

Sets or returns the position of a legend or title in a map frame.

Data type

Variant (MapPlacement enumeration)

Syntax

object.**Placement =** *value*

value **=** *object*.**Placement**

Legal values

Value	Description
$TopLeft	Place the legend or title in the top left.
$CenterLeft	Place the legend or title in the center left.
$BottomLeft	Place the legend or title in the bottom left.
$BottomCenter	Place the legend or title in the bottom center.
$BottomRight	Place the legend or title in the bottom right.
$CenterRight	Place the legend or title in the center right.
$TopRight	Place the legend or title in the top right.
$TopCenter	Place the legend or title in the top center.
$Custom	Place the legend or title in a non-standard position.

1-2-3: PlotPosition property

{button ,AL(`H_123_PLOT_CLASS;',0)} See list of classes

Sets or returns the manner in which a map plot is positioned.

Data type

Variant (PlotPosition enumeration)

Syntax

mapplot.**PlotPosition =** *value*

value **=** *mapplot*.**PlotPosition**

Legal values

Value	Description
$Default	(Default) The map uses the default plot positioning.
$Custom	Allows you to specify the plot positioning for the map.

1-2-3: PlotRotation property

{button ,AL(`H_123_PLOT_CLASS;',0)} See list of classes

Sets and returns the rotation of the map in degrees.

Data type

Long

Syntax

mapplot.**PlotRotation =** *value*

value **=** *mapplot*.**PlotRotation**

Legal values

The value for the PlotRotation property is any long between -360 to 360. The default value is 0 (zero).

1-2-3: Plot property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS',0)} See list of classes

(Read-only) Returns the plot object for a map.

Data type

MapPlot

Syntax

Set *mapplot* = *object*.**Plot**

Legal values

The value for the Plot property is the MapPlot object for a map.

1-2-3: PointCount property

{button ,AL(`H_123_DRAWLINE_CLASS;H_123_FREEHAND_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS',0)} See list of classes

Sets and returns the number of points in a Freehand drawing, line, polyline, or polygon.

Data type

Long

Syntax

value = *object*.**PointCount**

Legal values

The value for the PointCount property is the number of points in a Freehand drawing, line, polyline, or polygon.

1-2-3: PrinterNames property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_PRINTERNAMES_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns a collection of the printer names the application can use.

Data type

Strings

Syntax

Set *strings* **=** *application*.**PrinterNames**

Legal values

The value for the PrinterNames property is a collection of all the printer names the application can use.

```
' Example: PrinterNames property and Print method
' Print a list of the printers the application can use in the Output panel.
Forall x In CurrentApplication.PrinterNames
Print x
End Forall
```

1-2-3: PrinterName property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_PRINTERNAME_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the name of the currently selected printer.

Data type

String

Syntax

application.**PrinterName =** *value*

value **=** *application*.**PrinterName**

Legal values

The value of the PrinterName property is the name you set or the name of the currently selected printer.

```
' Example: CurrentPrintSettings and PrinterName properties
' Select a range, preview it, get the printer name, print the printer name
' to output, then print the range, and close preview window.
Dim prntnm As String
Set CurrentDocument.CurrentPrintSettings.PrintRange = [A:B5..A:H25]
CurrentApplication.Preview
prntnm=CurrentApplication.PrinterName
[A:B5].Contents = prntnm
CurrentApplication.Print
CurrentApplication.ClosePreview
```

1-2-3: PrinterQuality property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the print quality for the specified printer.

Data type

String

Syntax

application.**PrinterQuality =** *value*

value **=** *application*.**PrinterQuality**

Legal values

The default value for the PrinterQuality property is set in the Windows Control Panel.

1-2-3: PrintPagesFrom property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the number of the first page to print, with the value of 1 representing the first page in the selected range or collection.

Data type

Long

Syntax

printsettings.**PrintPagesFrom =** *value*

value **=** *printsettings*.**PrintPagesFrom**

Legal values

The value for the PrintPagesFrom property is the number of the first page you want to print.

1-2-3: PrintPagesStart property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the number to assign to the first page printed from the selected range or collection.

Data type

Long

Syntax

printsettings.**PrintPagesStart =** *value*

value **=** *printsettings*.**PrintPagesStart**

Legal values

The value for the PrintPagesStart property is the number assigned to the first page printed from the selected range or collection.

1-2-3: PrintPagesTo property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the number of the last page to print, beginning with the value of 1 representing the first page in the selected range or collection.

Data type

Long

Syntax

printsettings.**PrintPagesTo =** *value*

value **=** *printsettings*.**PrintPagesTo**

Legal values

The value for the PrintPagesTo property is the number of the last page you want to print.

1-2-3: PrintRangeSaved property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether to save the print selection with the named print styles and let you print the same range with the same settings each time.

Data type

Variant (Boolean)

Syntax

printsettings.**PrintRangeSaved =** *value*

value **=** *printsettings*.**PrintRangeSaved**

Legal values

Value	Description
TRUE	Save the print range with the named print styles.
FALSE	(Default) Do not save the print range with the named print styles.

1-2-3: PrintRange property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the range or collection of ranges to be printed.

Data type

Range or Ranges

Syntax

Set *printsettings*.**PrintRange =** *object*

Set *object* **=** *printsettings*.**PrintRange**

Legal values

The value for the PrintRange property is any on-sheet Range object or Ranges collection.

1-2-3: PrintWhat property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the range to print.

Data type

Variant (PrintWhat enumeration)

Syntax

printsettings.**PrintWhat =** *value*

value **=** *printsettings*.**PrintWhat**

Legal values

Value	Description
$AllSheets	Print all sheets.
$CurrentSelection	Print the currently selected range.
$CurrentSheet	Print the current sheet.
$None	No print range is stored as a print setting.

1-2-3: ProductVersion property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_PRODUCTVERSION_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns the release number of the product.

Data type

String

Syntax

value **=** *application*.**ProductVersion**

Legal values

The value of the ProductVersion property is a string containing the release number of the product.

```
' Example: ProductVersion property
' Display product version in a message box.
Messagebox ("The product version is " + CurrentApplication.ProductVersion)
```

1-2-3: Properties property

{button ,AL(`H_123_CLASSINFO_CLASS;',0)} See list of classes

(Read-only) Returns a collection of property names that the object accepts.

Data type

Strings

Syntax

Set *strings* **=** *classinfo*.**Properties**

Legal values

The value for the Properties property is a collection of strings containing property names that the object accepts.

1-2-3: QueryTables property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_QUERYTABLES_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns a collection of all the existing query table connections to Approach.

Data type

QueryTables

Syntax

Set *querytables* = *document*.**QueryTables**

Legal values

The value for the QueryTables property is a collection.

```
' Example: QueryTables property
' List in column A the current names of all query tables in a file.
' The file specified in the next line must be modified to point to your file.
Const QTFILE = "c:\lotus\work\123\qtfile.123"
Dim myfile As Document
Dim qtables As QueryTables
' Open a file with Query Tables
Set myfile = CurrentApplication.OpenDocument(QTFILE, "", "", "", True)
' Get all Query Table names
Set qtables = myfile.QueryTables
' Move to cell A1
.MoveCellPointer $Home, 1
' Show all Query Table names in column A
Forall X In qtables
' Put name in cell
Selection.Contents = X.Name
' Move down one row
.MoveCellPointer $Down, 1
End Forall
```

1-2-3: RangeHeaderInSort property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_RANGEHEADERINSORT_PROPERTY_EXSCRIPT ',1)} See example

Specifies whether there are column header labels in the range of data to sort.

Data type

Variant (Boolean)

Syntax

document.**RangeHeaderInSort =** *value*

value **=** *document*.**RangeHeaderInSort**

Legal values

Value	Description
TRUE	There are column headers in the range selected to sort. Do not sort the column headers with the data.
FALSE	There are no column headers in the range selected to sort. Sort the entire selection.

```
' Example: RangeHeaderInSort and RangeSortHeaderDepth properties
' Indicate that there are column header rows in the sort selection, so that they
' are not be sorted. The column header rows are set by the RangeSortHeaderDepth
property.
CurrentDocument.RangeHeaderInSort = True
CurrentDocument.RangeSortHeaderDepth = 2
```

1-2-3: RangeSortHeaderDepth property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_RANGESORTHEADERDEPTH_PROPERTY_EXSCRIPT;H_123_RANGEHEADERINSORT_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the number of rows in the sort header.

Data type

Long

Syntax

document.**RangeSortHeaderDepth =** *value*

value **=** *document*.**RangeSortHeaderDepth**

Legal values

The value for the RangeSortHeaderDepth property is the number of rows in the sort header.

```
' Example: RangeHeaderInSort, RangeSortHeaderDepth properties
' Indicate that the column header should not be sorted.
' The column header area is the number of rows set by the RangeSortHeaderDepth
property.
CurrentDocument.RangeHeaderInSort = TRUE
CurrentDocument.RangeSortHeaderDepth = 2
```

1-2-3: Ranges property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_RANGES_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns a collection of ranges.

Data type

Ranges

Syntax

Set *ranges* **=** *document*.**Ranges**

Legal values

The value for the Ranges property is a collection of ranges.

```
' Example: Ranges property
Dim r as Range
Set r = CurrentDocument.Ranges("A:A1")
```

1-2-3: ReadOnly property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_READONLY_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Determines whether the file is read-only or not.

Data type

Variant (Boolean)

Syntax

value **=** *document***.ReadOnly**

Legal values

Value	Description
TRUE	The file is read-only.
FALSE	(Default) The file is read-write.

```
' Example: ReadOnly property
' Find out if current file is read only.
Dim isread As Variant
isread = CurrentDocument.ReadOnly
Print isread
```

1-2-3: RecordsLimited property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Determines whether the number of records in a query table are limited or not.

Data type

Variant (Boolean)

Syntax

dataquery.**RecordsLimit =** *value*

value **=** *dataquery*.**RecordsLimited**

Legal values

Value	Description
TRUE	Limit the number of records retrieved.
FALSE	(Default) Do not limit the number of records retrieved.

1-2-3: RecordsLimitMax property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Sets or returns the maximum number of records that will be retrieved when creating a query table.

Data type

Long

Syntax

dataquery.**RecordsLimitMax =** *value*

value **=** *dataquery*.**RecordsLimitMax**

Legal values

The value for the RecordsLimitMax property is any long from 1 - 8191.

Usage

The RecordsLimitMax property is used only if the RecordsLimited property is TRUE.

1-2-3: Red property

{button ,AL(`H_123_COLOR_CLASS',0)} See list of classes

{button ,AL(`H_123_RED_PROPERTY_EXSCRIPT ',1)} See example

(Read-only) Returns the red value of an object.

Data type

Long

Syntax

value **=** *color*.**Red**

Legal values

The value of the Red property is any long 0 - 255, where 0 represents no red and 255 represents the most amount of red.

```
' Example: Red property
' Check the red value for the specified color.
Dim somecolor As Color
Dim redvalue As Long
Set somecolor = CurrentApplication.Colors(22)
redvalue = somecolor.Red
Print redvalue
```

1-2-3: RegionRange property

{button ,AL(`H_123_MAP_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS;',0)} See list of classes

Sets or returns a map data range that contains all the region names to be mapped.

Data type

Range

Syntax

object.**RegionRange =** *range*

range **=** *object*.**RegionRange**

Legal values

The default value for the RegionRange property is the range you set or the range you specified using Map - Ranges.

1-2-3: RegisteredCompany property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the name of the registered company for the software.

Data type

String

Syntax

value = *application*.**RegisteredCompany**

Legal values

The value for the RegisteredCompany property is a string containing a name.

1-2-3: RegisteredUser property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns the name of the registered user for the software.

Data type

String

Syntax

value = *application*.**RegisteredUser**

Legal values

The value for the RegisteredUser property is a string containing a name.

1-2-3: ReplaceString property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_REPLACE_METHOD_EXSCRIPT',1)} See example

Sets or returns the string of characters to replace the text or numbers you find using a Replace or ReplaceAll method.

Data type

String

Syntax

application.**ReplaceString =** *value*

value **=** *application*.**ReplaceString**

Legal values

The value for the ReplaceString property is the string of characters to replace the text or numbers you find using a Replace or ReplaceAll method.

1-2-3: RestrictOutput property

{button ,AL(`H_123_QUERYTABLE_CLASS',0)} See list of classes

Determines whether to restrict the results of a query to a specified output range.

Data type

Variant (Boolean)

Syntax

querytable.**RestrictOutput =** *value*

value **=** *querytable*.**RestrictOutput**

Legal values

Value	Description
TRUE	Restrict the results of a query.
FALSE	(Default) Do not restrict the results of a query.

1-2-3: Revisions property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_REVISIONS_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the revision history for the specified file.

Data type

String

Syntax

document.**Revisions =** *value*

value **=** *document*.**Revisions**

Legal values

The default value of the Revisions property is the text you set or the revision history you entered using File - Workbook Properties (General tab).

```
' Example Revisions property
' Make a new document, set revison history information, then display it in message box.
Dim testdoc As Document
Set testdoc = CurrentApplication.NewDocument
testdoc.Revisions = "These are document revisions."
MessageBox testdoc.Revisions
```

1-2-3: Revs property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_REVS_PROPERTY_EXSCRIPT;',1)} See example

(Read-only) Returns the total number revisions associated with the specified file.

Data type

Long

Syntax

value **=** *document*.**Revs**

Legal values

The default value of the Revs property is the number of the times the file has been saved.

```
' Example: Revs property
' Display number of revisions for current file.
Dim mydoc As Document
Dim numbofrevisions As Long
Set mydoc = CurrentDocument
numbofrevisions = mydoc.Revs
MessageBox "The number of revisions is " & Str(numbofrevisions)
```

1-2-3: RGB property

{button ,AL(`H_123_COLOR_CLASS',0)} See list of classes

{button ,AL(`H_123_RGB_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the RGB (red, green, blue) value of the Color object.

Data type

Long

Syntax

color.**RGB =** *value*

value **=** *color*.**RGB**

Legal values

The value for the RGB property is the RGB value of the Color object.

```
' Example: BackColor, Background, and RGB properties
' Set the background color for a range and display a messagebox
' showing the RGB value of the color.
Dim z As Color
Dim a As Long
Set z = CurrentApplication.Colors("violet")
Set [A:A1..A:A10].Background.Backcolor = z
a = z.RGB
MessageBox Hex(a)
```

1-2-3: RightBorder property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_RIGHTBORDER_PROPERTY_EXSCRIPT',1)} See example

With the Style or Color property, sets or returns the style or color of the right border of a range.

Data type

RangeBorder

Syntax

Set *range*.**RightBorder =** *rangeborder*

range.**RightBorder.Style =** *value*

Set *range*.**RightBorder.Color =** *color*

Set *rangeborder* **=** *range*.**RightBorder**

value **=** *range*.**RightBorder.Style**

Set *color* **=** *range*.**RightBorder.Color**

Legal values

A RangeBorder object.

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

```
' Example: RightBorder property; Select method
' Select a range, then change the right border style to a dashed line.
[A:B3..A:C5].Select
Selection.RightBorder.Style = $DashBorder
```

1-2-3: RightMargin property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the right margin print setting, in twips.

Data type

Long

Syntax

printsettings.**RightMargin =** *value*

value **=** *printsettings*.**RightMargin**

Legal values

The default value for the RightMargin property is what you set or the right margin setting you specified using File - Preview & Page Setup (Margins).

Usage

Combined with the LeftMargin property, this value should not be greater than the width of the paper.

1-2-3: Rotation property

{button ,AL(`H_123_ARC_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS',0)} See list of classes

Sets or returns the rotation of the graphic object.

Data type

Long

Syntax

object.**Rotation =** *value*

value **=** *object*.**Rotation**

Legal values

The value for the Rotation property is any long from -360 - 360.

Usage

The Rotation property does not apply to the following:

button controls

charts

data tables

edit texts

embedded objects

linked objects

links

map annotations

maps

pictures

1-2-3: Rounded property

{button ,AL(`H_123_RECTANGLE_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_GROUP_CLASS;',0)} See list of classes

Sets or returns whether a rectangle has rounded corners.

Data type

Variant (Boolean)

Syntax

rectangle.**Rounded =** *value*

value **=** *rectangle*.**Rounded**

Legal values

Value	Description
TRUE	The rectangle has rounded corners.
FALSE	The rectangle does not have rounded corners.

1-2-3: RowFolding property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ROWFOLDING_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the folding direction of an outline.

Data type

Variant (FoldDir enumeration)

Syntax

sheet.**RowFolding =** *value*

value **=** *sheet*.**RowFolding**

Legal values

Value	Description
$ParentBefore	Fold the row so that the parent row is above its child rows.
$ParentAfter	(Default) Fold the row so that the parent row is below its child rows.

Usage

If you change the folding direction when an outline exists already, the existing outline information is deleted.

```
' Example: RowFolding property, RowOutlineVisible property, DemoteRow method,
' and Select method
[A].RowOutlineVisible = True
[A].RowFolding = $ParentBefore
[A:A17..A:IV18].Select
Selection.DemoteRow
```

1-2-3: RowHeight property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_ROWHEIGHT_PROPERTY_EXSCRIPT;H_123_PATTERN_PROPERTY_EXSCRIPT ',1)} See example

Sets or returns the row height for the specified range.

Data type

Long

Syntax

range.**RowHeight** = *value*

value = *range*.**RowHeight**

Parameters

None

Legal values

Any long from 0 - 255. Setting the row height to 0 (zero) hides the row.

```
' Example: RowHeight property
' Select a range and adjust its row height
[A:B5..A:H25].Select
[A:B5..A:H25].RowHeight = 20
```

1-2-3: RowOutlineVisible property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ROWOUTLINEVISIBLE_PROPERTY_EXSCRIPT',1)} See example

Sets or returns whether the outline frame is displayed to the left of the row numbers.

Data type

Variant (Boolean)

Syntax

sheet.**RowOutlineVisible** = *value*

value = *sheet*.**RowOutlineVisible**

Legal values

Value	Description
TRUE	Display the outline frame to the left of the row numbers.
FALSE	Do not display the outline frame to the left of the row numbers.

```
' Example: RowOutlineVisible property; Select method
' Select a range and make the row outline visible.
[A:C8..A:C9].Select
[A].RowOutlineVisible = True
```

1-2-3: RowTitleRange property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the range of rows to be printed above the print range on each page of output.

Data type

Range

Syntax

printsettings.**RowTitleRange =** *value*

value **=** *printsettings*.**RowTitleRange**

Legal values

The value for the RowTitleRange property is a valid range, for example [A:A1..A:F1].

1-2-3: ScalePercent property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns how much the printed selection is shrunk or expanded.

Data type

Long

Syntax

printsettings.**ScalePercent =** *value*

value **=** *printsettings*.**ScalePercent**

Legal values

The default value of the ScalePercent property is any long from 15 - 1000. For example, if a print selection is to be shrunk to 75% of its normal size, the value would be 75.

1-2-3: SearchFormulas property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether to search for formulas when finding and replacing text and values.

Data type

Variant (Boolean)

Syntax

application.**SearchFormulas =** *value*

value **=** *application*.**SearchFormulas**

Legal values

Value	Description
TRUE	(Default) Search for formulas.
FALSE	Do not search for formulas.

1-2-3: SearchLabels property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether to search for labels when finding and replacing text and values.

Data type

Variant (Boolean)

Syntax

application.**SearchLabels =** *value*

value **=** *application*.**SearchLabels**

Legal values

Value	Description
TRUE	(Default) Search for labels.
FALSE	Do not search for labels.

1-2-3: SearchString property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the search string to use when finding and replacing text and values.

Data type

String

Syntax

application.**SearchString =** *value*

value **=** *application*.**SearchString**

Legal values

The value of the SearchString property is the search string used for finding and replacing text and values.

1-2-3: SearchValues property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether to search for numbers when finding and replacing text and values.

Data type

Variant (Boolean)

Syntax

application.**SearchValues =** *value*

value **=** *application*.**SearchValues**

Legal values

Value	Description
TRUE	(Default) Search for values.
FALSE	Do not search for values.

1-2-3: SelectFields property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

Sets or returns the list of field names to use to perform a query. When set, the query automatically updates to include these fields only.

Data type

String

Syntax

dataquery.**SelectFields =** *value*

value **=** *dataquery*.**SelectFields**

Legal values

The default value of the SelectFields property is the string that contains the field names in select order, delimited by semicolons.

Usage

The SelectFields property sets or returns original field names only; it does not recognize aliases or aggregates.

1-2-3: Selection property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SELECTION_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the currently selected objects in the active file.

Data type

Variant (object)

Syntax

Set *selectobjects* **=** *object***.Selection**

Legal values

The value of the Selection property is the currently selected object. If more than one object is selected, the value is a collection object that derives from the BaseCollection class.

{button ,AL(`H_123_CLIENT_VARIABLES_OVER',0)} See related topics

```
' Example:  Name and Selection properties; AddToSelection method
Sub TrySelection
   ' Create a new workbook.
   Dim file1 As Document
   Set file1 = CurrentApplication.NewDocument

   ' Select a range and show the range name in the message box.
   [A:A1].Select
   Messagebox "Current Selection: " + CurrentDocument.Selection.Name

   ' Create and select a rectangle and show the rectangle name in the message box.
   [A].NewRectangle 465,1530,1440,2760
   [Rectangle 1].Select
   Messagebox "Current Selection: " + CurrentDocument.Selection.Name

   ' Create and select a drawn line and show the line name in the message box.
   [A].NewDrawLine 240,450,975,975
   [Line 1].Select
   Messagebox "Current Selection: " + CurrentDocument.Selection.Name

   ' Create an ellipse and add it and the rectangle to the current selection.
   [A].NewEllipse 495,2610,1350,3165
   [Ellipse 1].AddToSelection
   [Rectangle 1].AddToSelection

   ' Change the edge color of the selected objects.
   Selection.EdgeColor.Colorname = "orchid"
End Sub
```

1-2-3: Share property

{button ,AL(`H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS',0)} See list of classes

Sets or returns the share options for a version or version group.

Data type

Variant (Share enumeration)

Syntax

object.**Share =** *value*

value **=** *object*.**Share**

Legal values

Value	Description
$Protected	Versions or version groups cannot be edited or deleted.
$Unprotected	(Default) Versions or version groups can be edited or deleted.
$Hidden	Versions or version groups cannot be edited, deleted, or seen.

Usage

The Share property applies to a locked file, and it does not override prior protection of a sheet or range.

1-2-3: SheetCount property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHEETCOUNT_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the number of sheets in the current file.

Data type

Long

Syntax

value **=** *document***.SheetCount**

Legal values

The value of the SheetCount property is any long from 0 - 255.

```
' Example: Name, SheetCount, Size, Subject, and Title properties
Sub DocStatistics
   ' This sub prints out information about the current document.

   Print "Name of workbook: " + CurrentDocument.Name
   Print "Number of sheets: " + Str(CurrentDocument.SheetCount)
   Print "Subject: " + CurrentDocument.Subject
   Print "Title: " + CurrentDocument.Title
   Print "File size: " + Str(CurrentDocument.Size)

End Sub
```

1-2-3: SheetDataPrint property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether sheet data is printed.

Data type

Variant (Boolean)

Syntax

printsettings.**SheetDataPrint =** *value*

value **=** *printsettings*.**SheetDataPrint**

Legal values

Value	Description
TRUE	(Default) Print sheet data.
FALSE	Do not print sheet data.

1-2-3: SheetFramePrint property

{button ,AL(`H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Determines whether row numbers and column letters will print in addition to sheet data.

Data type

Variant (Boolean)

Syntax

printsettings.**SheetFramePrint =** *value*

value **=** *printsettings*.**SheetFramePrint**

Legal values

Value	Description
TRUE	Print sheet frame.
FALSE	Do not print sheet frame.

1-2-3: SheetNumber property

{button ,AL(`H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_SHEETNUMBER_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the number of the sheet in a file, as opposed to the sheet letter or name.

Data type

Long

Syntax

value **=** *sheet*.**SheetNumber**

Legal values

The value of the SheetNumber property is any long from 0 - 255.

```
' Example: CurrentSheet, SheetNumber, and TabColor properties
Sub NewSheet
   ' Declare variables
   Dim Current As Sheet
   Dim SheetNum As Long

   ' Get the current sheet, and add another sheet after it.
   Set Current = CurrentDocument.CurrentSheet
   CurrentDocument.NewSheet $After,1,True

   ' Set the current sheet to be the sheet you just made, and select it.
   Set Current = CurrentDocument.CurrentSheet
   Current.Select

   ' Ask the user for a name for the sheet.
   Current.SheetName = Inputbox$("Name of new sheet?","Enter Sheet Name")

   ' Find out the number of the current sheet, and set the tab color
   ' based on the number; the first sheet is number 0.
   SheetNum = Current.SheetNumber
   Select Case SheetNum
     Case 1 : Selection.TabColor.ColorName = "pink"
     Case 2 : Selection.TabColor.ColorName = "red"
     Case 3 : Selection.TabColor.ColorName = "blue"
     Case Else : Selection.TabColor.ColorName = "yellow"
   End Select

End Sub
```

1-2-3: Sheets property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHEETS_PROPERTY_EXSCRIPT;H_123_ISZERODISPLAYED_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns a collection of all the sheets in the specified file.

Data type

Sheets

Syntax

Set *sheets* **=** *document***.Sheets**

Legal values

The value of the Sheets property is a Sheets object containing all the sheets in the specified file.

```
' Example: Count, Name, and Sheets properties; Item method
' This sub lists the sheets in the current document.
Sub ListSheets
   Dim numsheets As Long

   ' Find out how many sheets there are in the document.
   numsheets = CurrentDocument.Sheets.Count

   ' Cycle through the sheets, printing the name of each one.
   ' The first sheet in the collection is always number zero.
   For x = 0 To numsheets - 1
       Print CurrentDocument.Sheets.Item(x).Name
   Next
End Sub
```

1-2-3: ShortDayNames property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_SHORTDAYNAMES_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns a collection of strings that represent the abbreviated names of days.

Data type

Strings

Syntax

Set *strings* **=** *application***.ShortDayNames**

Legal values

The value of the ShortDayNames property is the collection of day names, which is determined by the country setting you specified in the Windows Control Panel.

```
' Example: ShortDayNames and ShortMonthNames properties; Item method
Sub EnterShortNames
   ' Print the short name of each day (7 in all).
   ' Add a space after each name, but don't put a line feed after each one
   ' (the line feed is suppressed by the semicolon).
   For x = 0 To 6
       Print CurrentApplication.ShortDayNames.Item(x) & " ";
   Next
   Print

   ' Print the short name of each month (12 in all).
   For x = 0 To 11
       Print CurrentApplication.ShortMonthNames.Item(x) & " ";
   Next
   Print
End Sub
' Output:
' Mon Tue Wed Thu Fri Sat Sun
' Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

1-2-3: ShortMonthNames property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_SHORTDAYNAMES_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns a collection of strings that represent the abbreviated names of months.

Data type

Strings

Syntax

Set *strings* **=** *application***.ShortMonthNames**

Legal values

The value of the ShortMonthNames property is the collection of month names, which is determined by the country setting you specified in the Windows Control Panel.

1-2-3: ShowAutomaticPageBreaks property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWPAGEBREAKS_PROPERTY_EXSCRIPT;H_123_GRIDLINECOLOR_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display automatic page breaks or not.

Data type

Variant (Boolean)

Syntax

object.**ShowAutomaticPageBreaks =** *value*

value **=** *object*.**ShowAutomaticPageBreaks**

Legal values

Value	Description
TRUE	(Default) Display automatic page breaks.
FALSE	Do not display automatic page breaks.

1-2-3: ShowCellCommentMarkers property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWMARKERS_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display cell comment markers.

Data type

Variant (Boolean)

Syntax

object.**ShowCellCommentMarkers =** *value*

value **=** *object*.**ShowCellCommentMarkers**

Legal values

Value	Description
TRUE	(Default) Display cell comment markers.
FALSE	Hide cell comment markers.

Usage

You can set cell comment markers to display in a file or in an application.

1-2-3: ShowDrawLayer property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWDRAWLAYER_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display graphic objects, including maps, charts, drawings, text blocks, buttons, and embedded objects.

Data type

Variant (Boolean)

Syntax

object.**ShowDrawLayer =** *value*

value **=** *object*.**ShowDrawLayer**

Legal values

Value	Description
TRUE	(Default) Display graphic objects.
FALSE	Hide graphic objects.

```
' Example: BackColor, Background, Color, ColorName, Pattern, and ShowDrawLayer
properties;
' NewChart method
Sub TestDraw
   CurrentApplication.NewDocument
   Dim mysheet As Sheet
   Set mysheet = CurrentDocument.CurrentSheet

   ' Create a rectangle.
   mysheet.NewRectangle 500,4000,3500,5000
   [Rectangle 1].Select
   Selection.Background.Pattern = $BowlingBalls
   Selection.Background.Color.ColorName = "blue"

   ' Add some data to the sheet.
   [A:A1].Contents = "45"
   [A:A2].Contents = "56"
   [A:A3].Contents = "92"
   [A:B1].Contents = "24"
   [A:B2].Contents = "35"
   [A:B3].Contents = "59"

   ' Create a chart.
   mysheet.NewChart 1020,1005,3165,3135,[A:A1..A:B3]

   Messagebox "Turn off display of drawn objects."
   mysheet.ShowDrawLayer = False

   Messagebox "Display drawn objects again."
   mysheet.ShowDrawLayer = True

End Sub
```

1-2-3: ShowFormulaMarkers property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWMARKERS_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display formula markers or not.

Data type

Variant (Boolean)

Syntax

object.**ShowFormulaMarkers =** *value*

value **=** *object*.**ShowFormulaMarkers**

Legal values

Value	Description
TRUE	Display formula markers.
FALSE	(Default) Hide formula markers.

1-2-3: ShowGridLines property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;H_123_GRIDLINECOLOR_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display grid lines in a sheet.

Data type

Variant (Boolean)

Syntax

object.**ShowGridLines =** *value*

value **=** *object*.**ShowGridLines**

Legal values

Value	Description
TRUE	(Default) Display grid lines.
FALSE	Hide grid lines.

1-2-3: ShowManualPageBreaks property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWPAGEBREAKS_PROPERTY_EXSCRIPT;H_123_GRIDLINECOLOR_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display manual page breaks.

Data type

Variant (Boolean)

Syntax

object.**ShowManualPageBreaks =** *value*

value **=** *object*.**ShowManualPageBreaks**

Legal values

Value	Description
TRUE	(Default) Display manual page breaks.
FALSE	Do not display manual page breaks.

```
' Example: Background, CellComment, Color, ColorName, GridLineColor, Pattern,
'  ShowCellCommentMarkers, ShowFormulaMarkers, ShowGridLines, and ShowSheetFrame
properties
Sub CustomLook
   ' Declare mysheet as the current sheet.
   Dim mysheet As Sheet
   Set mysheet = CurrentDocument.CurrentSheet

   ' Set sheet and gridline colors.
   mysheet.Background.Pattern = $SolidForeground
   mysheet.Background.Color.ColorName = "ivory"
   mysheet.GridLineColor.ColorName = "khaki"
   ' Note that ShowGridLines is true by default; this turns them on if the user
   ' had turned them off.
   mysheet.ShowGridLines = True
   mysheet.ShowSheetFrame = False

   ' Enter some values in the sheet
   [a:b2].Contents = "Values"
   [a:b3].Contents = "45"
   [a:b4].Contents = "84"
   [a:b5].Contents = "73"
   [a:a6].Contents = "Average"
   ' Enter a formula
   [a:b6].Contents = "@AVG(A:b2..A:b4)"
   ' Enter a cell note
   [a:b2].CellComment = "Values from data collected in October."

   ' Show all cell comment and formula markers.
   CurrentDocument.ShowFormulaMarkers = True
   ' Note that ShowCellCommentMarkers is true by default; this turns them on if the
   user
   ' had turned them off.
   CurrentDocument.ShowCellCommentMarkers = True
End Sub
```

```
' Example: HorizontalPageBreak, ShowAutomaticPageBreaks, ShowManualPageBreaks,
' and VerticalPageBreak properties; Print method
Sub MyPrint
   ' This example hides all page breaks, sets manual page breaks,
   ' prints the sheet, removes the manual page breaks, and then shows the breaks
   again.

   ' Hide page breaks.
   CurrentDocument.ShowManualPageBreaks = False
   CurrentDocument.ShowAutomaticPageBreaks = False

   ' Set manual page breaks.
   [A:d20].HorizontalPageBreak = True
   [A:d20].VerticalPageBreak = True

   ' Print the current sheet.
   CurrentDocument.CurrentPrintSettings.PrintWhat = $CurrentSheet
   CurrentApplication.Print

   ' Remove manual page breaks.
   [A:d20].HorizontalPageBreak = False
   [A:d20].VerticalPageBreak = False

   ' Show page breaks again.
   CurrentDocument.ShowManualPageBreaks = True
   CurrentDocument.ShowAutomaticPageBreaks = True

End Sub
```

1-2-3: ShowScrollBars property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWSHEETTABS_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display scroll bars in the specified file.

Data type

Variant (Boolean)

Syntax

object.**ShowScrollBars =** *value*

value **=** *object*.**ShowScrollBars**

Legal values

Value	Description
TRUE	(Default) Display scroll bars.
FALSE	Hide scroll bars.

1-2-3: ShowSheetFrame property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWMARKERS_PROPERTY_EXSCRIPT;H_123_SHOWSHEETTABS_PROPERTY_EXSCRIPT;H_123_GRIDLINECOLOR_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display the frame of the sheet.

Data type

Variant (Boolean)

Syntax

object.**ShowSheetFrame =** *value*

value **=** *object*.**ShowSheetFrame**

Legal values

Value	Description
TRUE	(Default) Display the sheet frame.
FALSE	Hide the sheet frame.

1-2-3: ShowSheetTabs property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWSHEETTABS_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display all the tabs for a sheet.

Data type

Variant (Boolean)

Syntax

object.**ShowSheetTabs =** *value*

value **=** *object*.**ShowSheetTabs**

Legal values

Value	Description
TRUE	(Default) Display all sheet tabs.
FALSE	Hide all sheet tabs.

```
' Example: ShowSheetFrame, ShowSheetTabs, and ShowScrollBars properties
Sub Biggest
   Dim answer As Long

   ' First restore the application window if it's maximized.
   CurrentApplication.ApplicationWindow.Restore

   ' Make current document take up the whole screen, and don't show sheet tabs,
   ' scroll bars, the sheet frame, the status bar, the edit line, or icons.
   CurrentApplication.ApplicationWindow.Maximize
   CurrentDocument.ShowSheetTabs = False
   CurrentDocument.ShowScrollBars = False
   CurrentDocument.ShowSheetFrame = False
   CurrentApplication.ApplicationWindow.StatusBarVisible = False
   CurrentApplication.ApplicationWindow.IconBarsVisible = False
   CurrentApplication.ApplicationWindow.EditLineVisible = False

   ' The "1" in the following statement means to use an OK/Cancel message box.
   answer = Messagebox("Click OK to show document at regular size",1)
   if answer = 1 then
     ' Restore the document to its original size, and turn on display
     ' of sheet tabs, scroll bars, the sheet frame, the status bar, and icons.
     CurrentDocument.ShowSheetTabs = True
     CurrentDocument.ShowScrollBars = True
     CurrentDocument.ShowSheetFrame = True
     CurrentApplication.ApplicationWindow.StatusBarVisible = True
     CurrentApplication.ApplicationWindow.IconBarsVisible = True
     CurrentApplication.ApplicationWindow.Restore
   End If
End Sub
```

1-2-3: ShowVersionBorders property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHOWVERSIONBORDERS_PROPERTY_EXSCRIPT',1)} See example

Determines whether to display version borders.

Data type

Variant (Boolean)

Syntax

object.**ShowVersionBorders =** *value*

value **=** *object*.**ShowVersionBorders**

Legal values

Value	Description
TRUE	(Default) Display version borders.
FALSE	Hide version borders.

```
'  Example: Contents, ShowVersionBorders, and VersionBorderVisible properties; Clear,
' CreateRangeName, and Select methods
Sub ShowBorders
   CurrentApplication.NewDocument
   ' Turn off display of version names and borders for this document.
   CurrentDocument.ShowVersionBorders = False

   ' Put some data in the sheet.
   [a:b5].Contents = "Low estimate"
   [a:b6].Contents = "12"
   [a:b7].Contents = "15"
   [a:b8].Contents = "17"
   [a:b9].Contents = "19"
   [a:c6].Contents = "14"
   [a:c7].Contents = "16"
   [a:c8].Contents = "19"
   [a:c9].Contents = "22"

   ' Create a new version.
   [A:b5..a:c9].Select
   CurrentDocument.CreateRangeName "DATA",[A:b5..a:c9]
   Selection.NewVersion "Version 1"

   ' Specify that it's OK to show the version border for this particular version.
   ' This statement does not actually turn the borders on.
   Selection.VersionBorderVisible = True
   Selection.Clear

   ' Add data for 2nd version.
   [a:b5].Contents = "High estimate"
   [a:b6].Contents = "15"
   [a:b7].Contents = "19"
   [a:b8].Contents = "24"
   [a:b9].Contents = "27"
   [a:c6].Contents = "18"
   [a:c7].Contents = "21"
   [a:c8].Contents = "25"
   [a:c9].Contents = "29"

   Messagebox "Turning on version borders again"
   CurrentDocument.ShowVersionBorders = True
End Sub
```

1-2-3: Size property

{button ,AL(`H_123_DOCUMENT_CLASS;H_123_FONT_CLASS;H_123_OLEOBJECT_CLASS;H_123_APPROACH CONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_QUERYTABLE_CLASS',0)} See list of classes

{button ,AL(`H_123_SHEETCOUNT_PROPERTY_EXSCRIPT;H_123_STYLENAME_PROPERTY_EXSCRIPT;H_123 _FONTNAME_PROPERTY_EXSCRIPT;H_123_FONT_PROPERTY_EXSCRIPT;H_123_ITALIC_PROPERTY_EX SCRIPT ',1)} See example

For Font objects: Sets or returns the point size of the font.

For Document objects: (Read-only) Returns the size of a file, in bytes.

For embedded or linked-to objects: (Read-only) Returns the size of the file in kilobytes.

Data type

For Font objects: Double

For Document objects: Long

For embedded or linked-to objects: Long

Syntax

For Font objects:

*font***.Size =** *value*

value **=** *font***.Size**

or

For Document objects:

value **=** *document***.Size**

or

For embedded or linked-to objects:

value **=** *object***.Size**

Legal values

For Font objects, the value of the Size property is the point size of the font.

For Document objects, the value of the Size property is the size of the file when last saved.

For embedded or linked-to objects, the value of the Size property is the size of the file in kilobytes.

1-2-3: SmartMasterOn property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether the New Workbook dialog box displays when you create a new file, letting you choose from a list of available SmartMaster templates.

Data type

Variant (Boolean)

Syntax

application.**SmartMasterOn =** *value*

value **=** *application*.**SmartMasterOn**

Legal values

Value	Description
TRUE	(Default) Display the New Workbook dialog box when creating a new file.
FALSE	Do not display the New Workbook dialog box when creating a new file.

1-2-3: SmartMasterPath property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the default path for SmartMaster templates.

Data type

String

Syntax

application.**SmartMasterPath =** *value*

value **=** *application*.**SmartMasterPath**

Legal values

The default value of the SmartMasterPath property is the path for SmartMaster templates you set or the path you specified using File - User Setup - 1-2-3 Preferences (File Locations tab). The install program sets the default path to \Lotus\SMasters\123.

1-2-3: SortDriver property

{button ,AL(`H_123_APPLICATION_CLASS;',0)} See list of classes

Sets or returns the country sort driver.

Data type

String

Syntax

application.**SortDriver =** *value*

value **=** *application*.**SortDriver**

Legal values

The value of the SortDriver property is the country sort driver you set or the country sort order you specified in the Windows Control Panel.

1-2-3: SortRange property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

Sets or returns the range that was last used by the Sort method.

Data type

Range

Syntax

Set *document*.**SortRange =** *range*

Set *range* **=** *document*.**SortRange**

Legal values

The default value of the SortRange property is the range that was last used by the Sort method.

1-2-3: SQL property

{button ,AL(`H_123_QUERY_CLASS',0)} See list of classes

{button ,AL(`H_123_SQL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the equivalent SQL statement for query operations.

Data type

String

Syntax

value **=** *dataquery***.SQL**

Legal values

The value of the SQL property is a string that represents the equivalent SQL statement for query operations.

```
' Example: SQL and DataQueryNames properties
' This sub prints the SQL strings for queries created with macro commands.
' You could use this sub to print the names of all the queries in your
' workbook, along with their SQL strings.
' Or you could use a similar sub to refresh the queries, using
' the Refresh method instead of the SQL property.
Sub PrintQueries
   Dim y As DataQuery

   ' Loop through all the queries in the current document.
   Forall x In CurrentDocument.DataQueryNames
       ' Print the name of the query (the semicolon prevents a newline).
       Print x,"  ";
       ' Assign a new query to the name specified in x.
       Set y = Bind(x)
       ' Print the SQL statement for the query.
       Print y.SQL
   End Forall

   Print
End Sub
```

1-2-3: Stapled property

{button ,AL(`H_123_PRINTSETTINGS_CLASS;',0)} See list of classes

Determines whether to staple a print selection after it has been printed.

Data type

Variant (Boolean)

Syntax

printsettings.**Stapled =** *value*

value **=** *printsettings*.**Stapled**

Legal values

Value	Description
TRUE	(Default) Staple the printed selection.
FALSE	Do not staple the printed selection.

Usage

This property applies only to printers with a stapling feature.

1-2-3: StartColumn property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTROWCOL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the value representing the column number that corresponds to the top left cell of a range.

Data type

Long

Syntax

value **=** *range***.StartColumn**

Legal values

The value of the StartColumn property is the zero-based column number of the top left cell of a range, from 0 - 255. The zero-based column number of column A is 0.

```
' Example: EndColumn, EndRow, EndSheet, StartColumn, StartRow, and StartSheet
properties
' This Sub checks whether the current selected range is exactly
' 4 rows long and 2 columns wide.
Sub CheckRangeShape

   Dim mysel As Range
   Dim flag As Integer

   ' Get the current selection.
   Set mysel = CurrentDocument.Selection

   ' Start by setting flag to false.
   flag = 0

   ' The number of columns in range must be 2.
   If mysel.EndColumn - mysel.StartColumn = 1 Then
       ' The number of rows in range must be 4.
       If mysel.EndRow - mysel.StartRow = 3 Then
           ' The entire range must be on one sheet.
           If mysel.EndSheet - mysel.StartSheet = 0 Then
               flag = 1
           End If
       End If
   End If

   ' Glag = 1 means that flag is set to true (range is correct size).
   If flag = 1 Then
       Messagebox "Range is correct size"
   Else
       Messagebox "Range is not correct size"
   End If

End Sub
```

1-2-3: StartRow property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTROWCOL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the row number associated with the address of the top left cell of the specified range.

Data type

Long

Syntax

value **=** *range***.StartRow**

Legal values

The value of the StartRow property is the zero-based row number of the top left cell of a range, from 0 - 8191. The zero-based row number of row 1 is 0.

1-2-3: StartSheet property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_STARTROWCOL_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the sheet number associated with the address of the top left cell of the specified range.

Data type

Long

Syntax

value **=** *range***.StartSheet**

Legal values

The value of the StartSheet property is the zero-based sheet number of the top left cell of a range, from 0 - 255. The zero-based sheet number of sheet A is 0.

1-2-3: StatusBarVisible property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS',0)} See list of classes

Determines whether the status bar is displayed.

Data type

Variant (Boolean)

Syntax

applicationwindow.**StatusBarVisible =** *value*

value **=** *applicationwindow*.**StatusBarVisible**

Legal values

Value	Description
TRUE	(Default) Display the status bar.
FALSE	Hide the status bar.

1-2-3: Strikethrough property

{button ,AL(`;H_123_FONT_CLASS',0)} See list of classes

{button ,AL(`H_123_STRIKETHROUGH_PROPERTY_EXSCRIPT',1)} See example

Determines whether data is styled using the strikethrough attribute.

Data type

Variant (Boolean)

Syntax

font.**Strikethrough =** *value*

value **=** *font*.**Strikethrough**

Legal values

Value	Description
TRUE	Apply the strikethrough style to data.
FALSE	(Default) Do not apply the strikethrough style to data.

```
'  Example: Bold, Font, and Strikethrough properties
Sub FixUpText
   ' Check whether current selection uses the strikethrough attribute.
   If CurrentDocument.Selection.Font.Strikethrough = True Then
       ' If so, then change it to bold.
       CurrentDocument.Selection.Font.Strikethrough = False
       CurrentDocument.Selection.Font.Bold = True
   End If

End Sub
```

1-2-3: StyleName property

{button ,AL(`;H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_STYLENAME_PROPERTY_EXSCRIPT',1)} See example

Sets or returns a named style for a range.

Data type

String

Syntax

range.**StyleName =** *value*

value **=** *range*.**StyleName**

Legal values

The value for the StyleName property is a string containing the name of an existing style.

```
'  Example: Bold, Font, FontColor, FontName, GridBorder, Size, and StyleName
properties;
' DefineNamedStyle method
Sub NameStyle
   Dim myrange As Range
   Set myrange = CurrentDocument.Selection

   ' Set the text styles and grid color of the current range.
   myrange.Font.FontName = "Lydian"
   myrange.Font.FontColor.ColorName = "manganese blue"
   myrange.Font.Size = 16
   myrange.Font.Bold = True
   myrange.GridBorder.Color.ColorName = "dk red violet"

   ' Define a style with the DefineNamedStyle method.
   myrange.DefineNamedStyle "MyStyle"

   ' Give this range that style name; you can also apply this style to other ranges.
   myrange.StyleName = "MyStyle"

End Sub
```

1-2-3: StyleRange property

{button ,AL(`;H_123_MAPBINS_CLASS',0)} See list of classes

Sets or returns the range specified for the color or pattern bins for a map.

Data type

Range

Syntax

Set *mapbins*.**StyleRange =** *range*

Set *range* **=** *mapbins*.**StyleRange**

Legal values

The default value of the StyleRange property is the range specified for the color or pattern bins for a map.

1-2-3: StyleSource property

{button ,AL(`;H_123_MAPBINS_CLASS',0)} See list of classes

Sets or returns the method used to set the values for colors and patterns bins for maps.

Data type

Variant (MapBinRenderSource enumeration)

Syntax

mapbins.**StyleSource =** *value*

value **=** *mapbins*.**StyleSource**

Legal values

Value	Description
$FromValues	(Default) 1-2-3 determines the values for colors and patterns bins.
$StylesManual	You manually set the values for colors and patterns bins.
$StylesFromRange	You specify the range that contains the values for colors and patterns bins.

1-2-3: StylesRetained property

{button ,AL(`H_123_VERSION_CLASS',0)} See list of classes

Determines whether to keep styles with versions.

Data type

Variant (Boolean)

Syntax

version.**StylesRetained =** *value*

value **=** *version*.**StylesRetained**

Legal values

Value	Description
TRUE	Keep styles with version.
FALSE	(Default) Do not keep styles with version.

Usage

You can create a style, for example, a background color, that is unique to each version. Or, you can keep the same styles for all versions of a range.

1-2-3: Style property

{button ,AL(`;H_123_RANGEBORDER_CLASS',0)} See list of classes

Sets or returns the border style for a range.

Data type

Variant (BorderStyleType enumeration)

Syntax

rangeborder.**Style =** *value*

value **=** *rangeborder*.**Style**

Legal values

The *rangeborder* can be any of the following: BottomBorder, GridBorder, HorizontalBorder, InnerBorder, LeftBorder, OutlineBorder, RightBorder, TopBorder, or VerticalBorder.

The following table shows the legal values for Style.

Value	Description
$NoBorder	(Default) Do not display a border around a range.
$SolidBorder	
$DoubleBorder	
$ThickBorder	
$DashBorder	
$DashSpaceBorder	
$LongDashBorder	
$DashDotBorder	
$DashDotDotBorder	

1-2-3: Subject property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SHEETCOUNT_PROPERTY_EXSCRIPT',1)} See example

Sets or returns subject information about the file.

Data type

String

Syntax

document.**Subject =** *value*

value **=** *document*.**Subject**

Legal values

The value of the Subject property is any valid string, with a maximum length of 256 characters.

1-2-3: SynchScrolling property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_SYNCHSCROLLING_PROPERTY_EXSCRIPT;H_123_EMBEDDED_PROPERTY_EXSCRIPT;', 1)} See example

Determines whether panes of a split window scroll individually or simultaneously.

Data type

Variant (Boolean)

Syntax

document.**SynchScrolling =** *value*

value **=** *document*.**SynchScrolling**

Legal values

Value	Description
TRUE	Scroll panes simultaneously.
FALSE	(Default) Do not scroll panes simultaneously.

```
' Example: SynchScrolling and ViewSplitStyle properties
Sub Split
   ' Split the window in half horizontally.
   CurrentDocument.ViewSplitStyle = $Horizontal

   ' Specify that both windows scroll simultaneously.
   CurrentDocument.SynchScrolling = True
End Sub
```

1-2-3: TabColor property

{button ,AL(`;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_TABCOLOR_PROPERTY_EXSCRIPT;H_123_SHEETNUMBER_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the color of the tab for the specified sheet.

Data type

Color

Syntax

Set *sheet*.**TabColor** = *color*

Set *color* = *sheet*.**TabColor**

Legal values

The value of the TabColor property is the color of the tab you set for the specified sheet or the tab color you specified using Sheet - Sheet Properties (Basics tab).

```
' Example: TabColor property
' Declare mysheet as the current sheet.
    Dim mysheet As Sheet
    Set mysheet = CurrentDocument.CurrentSheet
' Set the tab color by setting ColorName.
    mysheet.TabColor.ColorName = "red"
```

1-2-3: TabReturnKeyMovement property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the movement for the TAB and ENTER keys.

Data type

Variant (TabReturnKeyMovementChoices enumeration)

Syntax

application.**TabReturnKeyMovement** = *value*

value = *application*.**TabReturnKeyMovement**

Legal values

Value	Description
$DefaultMovement	TAB moves to the right one column. ENTER moves the cell pointer down.
$ClassicMovement	TAB moves right one screen. ENTER does not move the cell pointer.

1-2-3: TextCodepage property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns the character set when a file is read as text.

Data type

Variant (ReadCharSet enumeration)

Syntax

application.**TextCodepage =** *value*

value **=** *application*.**TextCodepage**

Legal values

Value	Description
$CP850	Multilinugal
$CP932	Japanese
$BIG5	Taiwanese
$KS	Korean
$GB	Chinese
$CP1252	US Windows
$CP437	US DOS
$CP860	Portugese
$CP863	French Canadian
$CP865	Norwegian/Danish
$CP1250	Eastern European Windows
$CP852	Eastern European DOS
$CP1251	Cyrillic Windows
$CP866	Cyrillic DOS
$CP1253	Greek Windows
$CP851	Greek DOS
$CP1254	Turkish Windows
$CP857	Turkish DOS
$CP1255	Hebrew Windows
$CP1256	Arabic Windows
$Windows	Windows ANSI
$DOS	DOS or OS/2

1-2-3: TextColumnParseOption property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns how a text file is parsed.

Data type

Variant (ReadTextAs enumeration)

Syntax

application.**TextColumnParseOption =** *value*

value **=** *application*.**TextColumnParseOption**

Legal values

Value	Description
$Tab	Use a tab as the delimiter.
$Comma	Use a comma as the delimiter.
$Semicolon	Use a semicolon as the delimiter.
$Space	Use a space as the delimiter.
$AutoParse	Use the layout of the file to determine how to parse the file.
$Other	Use a user-defined delimiter.
$None	No delimiter.
$Text	Text and numbers from a nondelimited file.
$Numbers	Text and numbers from a delimited file.

Usage

When the TextColumnParseOption property is set to $Other, you can use the TextColumnParseUserDefined property to set the delimiter.

{button ,AL(`H_123_TEXTCOLUMNPARSEUSERDEFINED_PROPERTY_MEMDEF',0)} See related topics

1-2-3: TextColumnParseUserDefined property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Sets or returns what delimiter to use to parse data while reading a file as text.

Data type

String

Syntax

application.**TextColumnParseUserDefined =** *value*

value **=** *application*.**TextColumnParseUserDefined**

Legal values

The value of the TextColumnParseUserDefined property is the delimiter to use to parse data while reading a file as text. The value can be a string containing up to three characters.

Usage

The TextColumnParseUserDefined property can be used only when the TextColumnParseOption property is set to $Other.

{button ,AL(`H_123_TEXTCOLUMNPARSEOPTION_PROPERTY_MEMDEF',0)} See related topics

1-2-3: TextHorizontalAlign property

{button ,AL(`;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_LEGEND_CLASS;H_123_MAPTITLE_CLASS;H_123_BUTTONCONTROL_CLASS',0)} See list of classes

{button ,AL(`H_123_TEXTXXX_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the horizontal alignment of data in a selection.

Data type

Variant (TextAlignment enumeration)

Syntax

object.**TextHorizontalAlign =** *value*

value **=** *object*.**TextHorizontalAlign**

Legal values

Value	Description
$AlignGeneral	Left-align text and right-align values in a selection.
$AlignLeft	Left-align data in a selection.
$AlignRight	Right-align data in a selection.
$AlignCenter	Center-align data in a selection.
$AlignEvenlySpaced	Evenly align text and labels with both the left and right margins in a selection.
$AlignRepeat	Repeat one or more characters across a cell

1-2-3: TextOrientation property

{button ,AL(`H_123_BUTTONCONTROL_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAPTITLE_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS;',0)} See list of classes

{button ,AL(`H_123_TEXTXXX_PROPERTY_EXSCRIPT',1)} See example

Sets or returns how text is oriented.

Data type

Variant (TextOrientation enumeration)

Syntax

object.**TextOrientation =** *value*

value **=** *object*.**TextOrientation**

Legal values

Value	Description
$OrientLeftToRight	
$OrientVertical	
$OrientUp	
$OrientDown	
$OrientRotate	

1-2-3: TextRotation property

{button ,AL(`;H_123_BUTTONCONTROL_CLASS;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_LEGEND_CLASS;H_123_MAPTITLE_CLASS',0)} See list of classes

{button ,AL(`H_123_TEXTXXX_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the rotation angle of data in a selection.

Data type

Long

Syntax

object.**TextRotation =** *value*

value **=** *object*.**TextRotation**

Legal values

The default value of the TextRotation property is 45.

1-2-3: TextVerticalAlign property

{button ,AL(`;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_LEGEND_CLASS;H_123_MAPTITLE_CLASS',0)} See list of classes

{button ,AL(`H_123_TEXTXXX_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the vertical alignment of data in a selection.

Data type

Variant (TextVertAlignment enumeration)

Syntax

object.**TextVerticalAlign =** *value*

value **=** *object*.**TextVerticalAlign**

Legal values

Value	Description
$AlignTop	Align text to the top of cells.
$AlignMiddle	Align text to the middle of cells.
$AlignBottom	Align text to the bottom of cells.

1-2-3: TextWrapped property

{button ,AL(`;H_123_EDITTEXT_CLASS;H_123_GROUP_CLASS;H_123_RANGE_CLASS;H_123_SHEET_CLASS;H_123_LEGEND_CLASS;H_123_MAPTITLE_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_DRAWCOLLECTION_CLASS;',0)} See list of classes

Determines whether data in a range is wrapped.

Data type

Variant (Boolean)

Syntax

object.**TextWrapped =** *value*

value **=** *object*.**TextWrapped**

Legal values

Value	Description
TRUE	Automatically wrap data in the range.
FALSE	(Default) Do not wrap data in the range.

```
'  Example: TextHorizontalAlign, TextOrientation, TextRotation, TextVerticalAlign, and
TextWrapped properties
'  Open a new document and call it testdocument.
     Dim testdocument As Document
     Set testdocument = CurrentApplication.NewDocument("testdocument")
'  Enter "Hello" in cell B1 and increase row height.
     [B1].Contents = "Hello"
     [B1].RowHeight = 48
'  Right-align the text.
     MessageBox("Right-align text.")
     [B1].TextHorizontalAlign = $AlignRight
'  Set orientation to up.
     MessageBox("Set orientation to up.")
     [B1].TextOrientation = $OrientUp
'  Set rotation to 60 degrees.
     MessageBox("Set rotation to 60 degrees.")
     'Note: You must set TextOrientation to $OrientRotate to use TextRotation.
     [B1].TextOrientation = $OrientRotate
     [B1].TextRotation = 60
' Set vertical alignment to middle.
     MessageBox("Set vertical alignment to middle.")
     [B1].TextOrientation = $OrientLeftToRight
     [B1].TextVerticalAlign = $AlignMiddle
'  Enter some text in cell B3 and wrap it within one cell.
     MessageBox("Wrap some text.")
     [B3].Contents = "My name is Allison. What is your name?"
     [B3].TextWrapped = True

'Use the MessageBox statement to display a
'message asking if you want to delete the test documents and test file.
     Dim boxtype As Long, answer As Integer
     boxtype& = 4 + 32
     '4 = MB_YESNO; 32 = MB_ICONQUESTION
     'Note: %INCLUDE LSCONST.LSS in your script declarations to use
     'the constants instead of the numbers with the MessageBox statement.
     answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
     If answer% = 6 Then
     'If the answer is 6 (IDYES), close the test document
         CurrentDocument.Close False
     End If
```

1-2-3: Text property

{button ,AL(`;H_123_BUTTONCONTROL_CLASS;H_123_EDITTEXT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPBIN_CLASS;',0)} See list of classes

{button ,AL(`H_123_TEXT_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the text string in a button, text block, map bin label, or map title.

Data type

String

Syntax

object.**Text =** *value*

value **=** *object*.**Text**

Legal values

The value of the Text property is the text string in the button, text block, map bin label, or map title.

```
'  Example: Text property; Clear and NewButton methods
'  Create a button and change the button text.
    Dim mysheet As Sheet
    Dim mybutton As ButtonControl
    Set mysheet = CurrentDocument.CurrentSheet
    MessageBox("Create a button.")
    Set mybutton = mysheet.NewButton(975,2205,3285,2655)
    MessageBox("Change the button text.")
    mybutton.Text = "Hello"
    MessageBox("Delete the button.")
    MyButton.Clear
```

1-2-3: ThousandsSeparator property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_THOUSANDSSEPARATOR_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the separator character used for numbers over 999.

Data type

String

Syntax

value **=** *application*.**ThousandsSeparator**

Legal values

The value of the ThousandsSeparator property is set in the Windows Control Panel.

```
'  Example: ThousandsSeparator property
'  Return the thousands separator character in a message box.
    Dim thousandssep As String
    thousandssep = CurrentApplication.ThousandsSeparator
    MessageBox("The thousands separator is " + thousandssep)
```

1-2-3: TimeSeparator property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_TIMESEPARATOR_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the type of separator used between hours and minutes, and minutes and seconds.

Data type

String

Syntax

value **=** *application*.**TimeSeparator**

Legal values

The default value of the TimeSeparator property is set in the Windows Control Panel.

```
'  Example: TimeSeparator property
'  Return the time separator character in a message box.
    Dim timesep As String
    timesep = CurrentApplication.TimeSeparator
    MessageBox("The time separator is " + TimeSep)
```

1-2-3: Title property

{button ,AL(`;H_123_DRAWCOLLECTION_CLASS;H_123_MAP_CLASS;H_123_DOCUMENT_CLASS;H_123_GROUP_CLASS;',0)} See list of classes

{button ,AL(`H_123_TITLE_PROPERTY_EXSCRIPT;H_123_SHEETCOUNT_PROPERTY_EXSCRIPT',1)} See example

Returns the MapTitle object for a specified map.

Sets or returns the title of a specified file.

Data type

For Map objects: MapTitle

For Document objects: String

Syntax

Set *maptitle* **=** *map*.**Title**

or

Set *document*.**Title =** *value*

Set *value* **=** *document*.**Title**

Legal values

For Map objects, the value of the Title property is a MapTitle object.

For Document objects, the value of the Title property is a string.

```
'  Example: Title property
'  Create a new document and name it.
'  Open a new document and call it testdocument.
    Dim testdocument As Document
    Set testdocument = CurrentApplication.NewDocument("testdocument")
'  Set the document title to "mydocument"
    testdocument.Title = "mydocument"
'  Display the title in a messagebox.
    MessageBox(TestDocument.Title)

'  Use the MessageBox statement to display a
'  message asking if you want to close the test document.
    Dim boxtype As Long, answer As Integer
    boxtype& = 4 + 32
    '4 = MB_YESNO; 32 = MB_ICONQUESTION
    'Note: %INCLUDE LSCONST.LSS in your script declarations to use
    'the constants instead of the numbers with the MessageBox statement.
    answer% = Messagebox("Do you want to close the test document
now?",boxType&,"Continue?")
    If answer% = 6 Then
    'If the answer is 6 (IDYES), close the test document
        CurrentDocument.Close False
    End If
```

1-2-3: TopBorder property

{button ,AL(`;H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_TOPBORDER_PROPERTY_EXSCRIPT',1)} See example

With the Style or Color property, sets or returns the style or color of the top border of a range.

Data type

RangeBorder

Syntax

Set *range*.**TopBorder.Style =** *value*

Set *range*.**TopBorder.Color =** *value*

Set *range*.**TopBorde =** *rangeborder*

Set *value* **=** *range*.**TopBorder.Style**

Set *value* **=** *range*.**TopBorder.Color**

Legal values

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

A RangeBorder object.

```
'  Example: TopBorder property
'  Add a solid, thick border style to the top border of the range.
    [B3].TopBorder.Style = $ThickBorder
```

1-2-3: TopMargin property

{button ,AL(`;H_123_PRINTSETTINGS_CLASS',0)} See list of classes

Sets or returns the top margin setting for the print selection, in twips.

Data type

Long

Syntax

printsettings.**TopMargin =** *value*

value **=** *print*.**TopMargin**

Legal values

The value for the TopMargin property is the top margin you set or the top margin you specified using File - Preview & Page Setup (Layout tab).

1-2-3: Top property

{button ,AL(`H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_QUERYTABLE_CLASS;H_123_RECTANGLE_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)} See list of classes

For a graphic object, sets or returns the vertical coordinate of the top handle.

For ApplicationWindow, DocWindow, or Window objects, sets or returns the top coordinate of the window.

Data type

Long

Syntax

object.**Top =** *value*

value **=** *object*.**Top**

Legal values

For graphic objects: The value of the Top property is the top boundary of a bounding rectangle, in units of twips.

For ApplicationWindow, DocWindow, and Window objects: The value for the Top property is the top coordinate of the window in pixels.

Usage

If the window is minimized or maximized, you will not see the effect of the Top property setting until the window is restored.

1-2-3: TotalMemory property

{button ,AL(`;H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns how much memory is available.

Data type

Long

Syntax

value **=** *application*.**TotalMemory**

Legal values

The value of the TotalMemory property is the amount of memory available in bytes.

1-2-3: Underline property

{button ,AL(`;H_123_FONT_CLASS',0)} See list of classes

{button ,AL(`H_123_UNDERLINE_PROPERTY_EXSCRIPT',1)} See example

Determines whether data is styled using the underline attribute.

Data type

Variant (Boolean)

Syntax

font.**Underline** = *value*

value = *font*.**Underline**

Legal values

Value	Description
TRUE	Apply the underline attribute to data.
FALSE	(Default) Do not apply the underline attribute to data.

```
'  Example: Underline property
'  Add the underline attribute to text in in the range.
    [A1].Font.Underline = True
```

1-2-3: UndoEnabled property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_UNDOENABLED_PROPERTY_EXSCRIPT',1)} See example

Determines whether Undo is enabled.

Data type

Variant (Boolean)

Syntax

application.**UndoEnabled =** *value*

value **=** *application*.**UndoEnabled**

Legal values

Value	Description
TRUE	(Default) Undo is enabled.
FALSE	Undo is not enabled.

```
'  Example: UndoEnabled property
'  Get the value of UndoEnabled and display a message box telling if it is enabled or
not.
    If CurrentApplication.UndoEnabled = True Then
        MessageBox("Undo is enabled.")
    Else
        MessageBox("Undo is not enabled.")
    End If
```

1-2-3: UpdateLinksOnOpenDoc property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_UPDATELINKSONOPENDOC_PROPERTY_EXSCRIPT',1)} See example

Determines whether links are updated when a file is opened.

Data type

Variant (Boolean)

Syntax

application.**UpdateLinksOnOpenDoc =** *value*

value **=** *application*.**UpdateLinksOnOpenDoc**

Legal values

Value	Description
TRUE	(Default) Update links when a file is opened.
FALSE	Do not update links when a file is opened.

```
'  Example: UpdateLinksOnOpenDoc property
'  Get the value for link updating, and display a message box about it.
    If CurrentApplication.UpdateLinksOnOpenDoc = True Then
        Messagebox("Link updating is on.")
    Else
        Messagebox("Link updating is off.")
    End If
```

1-2-3: UseOSDefaultColors property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

Determines whether to use the operating system default text and background colors for new workbook files.

Data type

Variant (Boolean)

Syntax

application.**UseOSDefaultColors** = *value*

value = *application*.**UseOSDefaultColors**

Parameters

None

Legal values

Value	Description
TRUE	Use the operating system default text and background colors. This sets the DefaultTextColor and DefaultBackColor properties to the operating system defaults.
FALSE	Use the default text and background colors set by the DefaultTextColor and DefaultBackColor properties.

1-2-3: UserClassNameApplication property

{button ,AL(`;H_123_APPROACHCONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

(Read-only) Returns the name of the application.

Data type

String

Syntax

appname **=** *object***.UserClassNameApplication**

Legal values

The value of the UserClassNameApplication property is the name of the application.

1-2-3: UserClassNameFull property

{button ,AL(`;H_123_APPROACHCONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

(Read-only) Returns the full name of the specified object, as registered by the server.

Data type

String

Syntax

fullname **=** *object*.**UserClassNameFull**

Legal values

The value of the UserClassNameFull property is the full name of the object.

1-2-3: UserClassNameShort property

{button ,AL(`;H_123_APPROACHCONNECTION_CLASS;H_123_DATALINK_CLASS;H_123_OLEOBJECT_CLASS;
H_123_QUERYTABLE_CLASS',0)} See list of classes

(Read-only) Returns the short name of the specified object, as registered by the server.

Data type

String

Syntax

shortname **=** *object***.UserClassNameShort**

Legal values

The value of the UserClassNameShort property is the short name of the object.

1-2-3: UserName property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_USERNAME_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the name of the user associated with the current computer.

Data type

String

Syntax

value = *application*.**UserName**

Legal values

The value of the UserName property is the name of the user.

```
'  Example: UserName property
'  Return the name of the user in a message box.
    Dim username As String
    username = CurrentApplication.UserName
    MessageBox(UserName)
```

1-2-3: UsingTotalToAutoSum property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_USINGTOTALTOAUTOSUM_PROPERTY_EXSCRIPT',1)} See example

Determines whether typing "total" in a cell activates the AutoSum feature or not.

Data type

Variant (Boolean)

Syntax

application.**UsingTotalToAutoSum =** *value*

value **=** *application*.**UsingTotalToAutoSum**

Legal values

Value	Description
TRUE	(Default) Enable typing "total" to activate AutoSum.
FALSE	Do not enable typing "total" to activate AutoSum.

```
'  Example: UsingTotalToAutoSum property
'  Turn AutoTotal on.
    MessageBox("Turn on using ""Total"" to automatically calculate a sum.")
    CurrentApplication.UsingTotalToAutoSum = True
```

1-2-3: ValueRange property

{button ,AL(`;H_123_MAPBINS_CLASS',0)} See list of classes

Sets or returns the range that contains the data for the specified map bins.

Data type

Range

Syntax

Set *mapbins*.**ValueRange =** *range*

Set *range* **=** *mapbins*.**ValueRange**

Legal values

The value of the ValueRange property is the range that contains the data for the specified map bins.

1-2-3: ValueSource property

{button ,AL(`;H_123_MAPBINS_CLASS',0)} See list of classes

(Read-only) Returns the method used to set the values for the specified map data bins.

Data type

Variant (MapBinValueSource enumeration)

Syntax

value **=** *mapbins*.**ValueSource**

Legal values

Value	Description
$Computed	(Default) 1-2-3 determines the values.
$Manual	You manually specify the values
$FromRange	You specify the range that contains the values.

1-2-3: Value property

{button ,AL(`H_123_MAPBIN_CLASS',0)} See list of classes

Sets or returns the value of the specified map bin.

Data type

Variant

Syntax

mapbin.**Value =** *value*

value **=** *mapbin*.**Value**

Legal values

The value of the Value property is the value of the map bin you set or the value you specified using Map - Color Bins or Map - Pattern Bins.

1-2-3: VersionBorderVisible property

{button ,AL(`H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_VERSION_METHOD_EXSCRIPT ',1)} See example

Determines whether to show the version border for the specified range.

Data type

Variant (Boolean)

Syntax

range.**VersionBorderVisible =** *value*

value **=** *range*.**VersionBorderVisible**

Legal values

Value	Description
TRUE	(Default) Show the version border.
FALSE	Do not show the version border.

1-2-3: VersionId property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_ARC_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_BACKGROUND_CLASS;H_123_BASEOBJECT_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_CLASSINFO_CLASS;H_123_COLOR_CLASS;H_123_DATALINK_CLASS;H_123_DATETIME_CLASS;H_123_DOCUMENT_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FONT_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPBIN_CLASS;H_123_PLOT_CLASS;H_123_MAPTEXTENTRY_CLASS;H_123_MAPTITLE_CLASS;H_123_MENU_CLASS;H_123_MENUBAR_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_PRINTSETTINGS_CLASS;H_123_QUERY_CLASS;H_123_QUERYTABLE_CLASS;H_123_RANGE_CLASS;H_123_RANGEBORDER_CLASS;H_123_RANGESELECTOR_CLASS;H_123_RECTANGLE_CLASS;H_123_SHEET_CLASS;H_123_VERSION_CLASS;H_123_VERSIONGROUP_CLASS;H_123_WINDOW_CLASS;',0)} See list of classes

{button ,AL(`H_123_VERSION_METHOD_EXSCRIPT ',1)} See example

(Read-only) Returns which version of the class was in use when the specified object was last modified.

Data type

Long

Syntax

value **=** *object***.VersionId**

Legal values

The value of the VersionId property is which version of the class was in use when the specified object was last modified.

1-2-3: VersionStatus property

{button ,AL(`H_123_RANGE_CLASS;',0)} See list of classes

{button ,AL(`H_123_VERSIONSTATUS_PROPERTY_EXSCRIPT',1)} See example

(Read-only) Returns the version status of a range.

Data type

Variant (VersionStatus enumeration)

Syntax

value = *range*.**VersionStatus**

Parameters

None

Legal values

Value	Description
$NotVersioned	The ranges contains no versions.
$Versioned	The range contains versions.
$VersionBordered	The range contains versions, and the version border is visible. This is returned in conjunction with the value $Versioned.
$VersionedHiddenCurrent	The range contains versions, and they are hidden. This is returned in conjunction with the value $Versioned.
$MemberOfCollection	The range is a member of the current worksheet collection.
$MultiCellOverlapped	The range contains versions, and some cells overlap another range that contains versions.
$SingleCellOverlapped	The range contains versions, and one cell overlaps another range that contains versions.

```
'  Example: VersionStatus property
'  Return the version status of the range in a message box.
    Dim statusofA1C3 As Variant
    statusofA1C3 = [A1..C3].VersionStatus
    MessageBox(statusofA1C3)
```

1-2-3: VerticalBorder property

{button ,AL(`;H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_VERTICALBORDER_PROPERTY_EXSCRIPT',1)} See example

With the Style or Color property, sets or returns the style or color of the vertical border of a range.

Data type

RangeBorder

Syntax

Set *range*.**VerticalBorder.Style =** *value*

Set *range*.**VerticalBorder.Color =** *value*

Set *range*.**VerticalBorder =** *rangeborder*

Set *value* **=** *range*.**VerticalBorder.Style**

Set *value* **=** *range*.**VerticalBorder.Color**

Set *rangeborder* **=** *range*.**VerticalBorder**

Legal values

See the Style property for a list of border styles.

See the Color palette for a list of border colors.

A RangeBorder object.

```
'  Example: VerticalBorder property
'  Add a solid, thick border style to the vertical border of the range.
    [B10].VerticalBorder.Style = $ThickBorder
```

1-2-3: VerticalPageBreak property

{button ,AL(`;H_123_RANGE_CLASS',0)} See list of classes

{button ,AL(`H_123_VERTICALPAGEBREAK_PROPERTY_EXSCRIPT;H_123_SHOWPAGEBREAKS_PROPERTY_EXSCRIPT',1)} See example

Determines whether a vertical page break exists to the left of the leftmost column in the specified range.

Data type

Variant (Boolean)

Syntax

range.**VerticalPageBreak =** *value*

value **=** *range*.**VerticalPageBreak**

Legal values

Value	Description
TRUE	(Default) Insert the vertical page break.
FALSE	Delete the vertical page break.

```
'  Example: VerticalPageBreak property
'  Insert a vertical page break.
    MessageBox("Insert a vertical page break to the left of column D.")
    [D1].VerticalPageBreak = True
    MessageBox("Remove the vertical page break.")
    [D1].VerticalPageBreak = False
```

1-2-3: VerticalTitle property

{button ,AL(`;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_VERTICALTITLE_PROPERTY_EXSCRIPT;H_123_HORIZONTALTITLE_PROPERTY_EXSCRIPT',1)} See example

Determines whether the vertical titles are frozen, based on the current location of the cell pointer.

Data type

Variant (Boolean)

Syntax

sheet.**VerticalTitle =** *value*

value **=** *sheet*.**VerticalTitle**

Legal values

Value	Description
TRUE	(Default) Vertical titles are frozen.
FALSE	Vertical titles are not frozen.

```
'  Example: VerticalTitle property
'  Select a range and add a frozen title.
    Messagebox("Freeze titles in column A.")
    [B1].Select
    [].VerticalTitle = True
    [Z1].Goto
    Messagebox("Turn off frozen titles.")
    [].VerticalTitle = False
```

1-2-3: VerticalScrollBarVisible property

{button ,AL(`H_123_APPLICATIONWINDOW_CLASS;H_123_DOCWINDOW_CLASS;H_123_WINDOW_CLASS',0)}
See list of classes

(Read-only) Returns whether the vertical scroll bar is displayed or not.

Data type

Variant (Boolean)

Syntax

value **=** *object*.**VerticalScrollBarVisible**

Legal values

Value	Description
TRUE	(Default) Display the vertical scroll bar.
FALSE	Hide the vertical scroll bar.

1-2-3: ViewSplitHeight property

{button ,AL(`;H_123_DOCUMENT_CLASS',0)} See list of classes

Sets or returns the height in pixels of the upper pane of the current document window, when the window is split.

Data type

Long

Syntax

document.**ViewSplitHeight =** *value*

value **=** *document*.**ViewSplitHeight**

Legal values

The values for the ViewSplitHeight property depend on the display setting. The value must be at least the height of one row.

{button ,AL(`H_123_VIEWSPLITWIDTH_PROPERTY_MEMDEF;H_123_VIEWSPLITSTYLE_PROPERTY_MEMDEF',0)} See related topics

1-2-3: ViewSplitStyle property

{button ,AL(`H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_VIEWSPLITSTYLE_PROPERTY_EXSCRIPT;H_123_SYNCHSCROLLING_PROPERTY_EXSCRIPT;H_123_EMBEDDED_PROPERTY_EXSCRIPT;',1)} See example

Sets or returns the way the Sheet window is split for viewing.

Data type

Variant (WindowSplitType enumeration)

Syntax

document.**ViewSplitStyle =** *value*

value **=** *document*.**ViewSplitStyle**

Legal values

Value	Description
$Horizontal	Split the window pane horizontally, above the cell pointer.
$Vertical	Split the window pane vertically, to the left of the cell pointer.
$FourWay	Split the window pane horizontally and vertically, above and to the left of the cell pointer.
$NoSplits	Do not split the window pane.

```
'  Example: ViewSplitStyle property
'  Split the current window into 2 vertical panes.
    [D10].Select
    MessageBox("Split the window vertically.")
    .ViewSplitStyle = $Vertical
    MessageBox("Unsplit the window.")
    .ViewSplitStyle = $NoSplits
```

1-2-3: ViewSplitWidth property

{button ,AL(`;H_123_DOCUMENT_CLASS',0)} See list of classes

Sets or returns the width in pixels of the left pane of the current document window, when the window is split.

Data type

Long

Syntax

document.**ViewSplitWidth =** *value*

value **=** *document*.**ViewSplitWidth**

Legal values

The values for the ViewSplitWidth property depend on the display setting. The value must be at least the width of one column.

{button ,AL(`H_123_VIEWSPLITHEIGHT_PROPERTY_MEMDEF;H_123_VIEWSPLITSTYLE_PROPERTY_MEMDEF',0)} See related topics

1-2-3: Visible property

{button ,AL(`;H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS;H_123_WINDOW_CLASS;H_123_PLOT_CLASS',0)} See list of classes

Determines whether the specified object is currently displayed.

The Visible property is a read-only property if the specified object is the Application object.

Data type

Variant (Boolean)

Syntax

object.**Visible =** *value*

value **=** *object*.**Visible**

Legal values

Value	Description
TRUE	(Default) Display the object.
FALSE	Hide the object.

1-2-3: WelcomeOn property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

{button ,AL(`H_123_WELCOMEON_PROPERTY_EXSCRIPT',1)} See example

Determines whether the Welcome to 1-2-3 dialog box appears at startup or not.

Data type

Variant (Boolean)

Syntax

application.**WelcomeOn** = *value*

value = *application*.**WelcomeOn**

Legal values

Value	Description
TRUE	(Default) Display the Welcome to 1-2-3 dialog box at startup.
FALSE	Do not display the Welcome to 1-2-3 dialog box at startup.

```
' Example: WelcomeOn property
'  Return a message box saying whether the welcome to 1-2-3 dialog box gets displayed.
    Dim iswelcomeon As Variant
    iswelcomeon = CurrentApplication.WelcomeOn
    MessageBox("WelcomeOn is currently " + iswelcomeon + ".")
```

1-2-3: WideUnderline property

{button ,AL(`H_123_FONT_CLASS',0)} See list of classes

{button ,AL(`H_123_WIDEUNDERLINE_PROPERTY_EXSCRIPT',1)} See example

Determines whether data is styled using the wide underline attribute.

Data type

Variant (Boolean)

Syntax

font.**WideUnderline =** *value*

value **=** *font*.**WideUnderline**

Legal values

Value	Description
TRUE	(Default) Apply the wide underline attribute to data.
FALSE	Do not apply the wide underline attribute to data.

```
'  Example: WideUnderline property
'  Add the wide underline attribute to the range.
    [A1].Font.WideUnderline = True
```

1-2-3: Width property

{button ,AL(`;H_123_APPLICATION_CLASS;H_123_APPLICATIONWINDOW_CLASS;H_123_APPROACHCONNECTION_CLASS;H_123_ARC_CLASS;H_123_BUTTONCONTROL_CLASS;H_123_CHART_CLASS;H_123_DOCWINDOW_CLASS;H_123_DRAWCOLLECTION_CLASS;H_123_DRAWLINE_CLASS;H_123_DRAWOBJECT_CLASS;H_123_EDITTEXT_CLASS;H_123_ELLIPSE_CLASS;H_123_FREEHAND_CLASS;H_123_GROUP_CLASS;H_123_LEGEND_CLASS;H_123_MAP_CLASS;H_123_MAPTITLE_CLASS;H_123_OLEOBJECT_CLASS;H_123_PICTURE_CLASS;H_123_POLYGON_CLASS;H_123_POLYLINE_CLASS;H_123_RECTANGLE_CLASS;H_123_QUERYTABLE_CLASS;H_123_WINDOW_CLASS;H_123_PLOT_CLASS;',0)} See list of classes

{button ,AL(`H_123_WIDTH_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the width of the current window, or the width of the bounding box around a graphic object.

Data type

Long

Syntax

object.**Width =** *value*

value **=** *object*.**Width**

Legal values

The value of the Width property is the width of the current window, in pixels, or the width of the bounding box around a graphic object, in twips.

Usage

If the window is minimized or maximized, you will not see the effect of the Width property setting until the window is restored.

```
'  Example: Width property
'  Return the width of the current window in a message box.
    Dim winwidth As Long
    winwidth = CurrentWindow.Width
    MessageBox("The width of the current window is " + Cstr(winwidth) + ".")
```

1-2-3: WindowsDefaultsDisplayed property

{button ,AL(`;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_WINDOWS_DEFAULTS_DISPLAYED_PROPERTY_EXSCRIPT',1)} See example

Determines whether the sheet displays the default Windows colors for text and background.

Data type

Variant (Boolean)

Syntax

sheet.**WindowsDefaultsDisplayed** = *value*

value = *sheet*.**WindowsDefaultsDisplayed**

Legal values

Value	Description
TRUE	(Default) Display the default Windows colors.
FALSE	Display custom window colors.

```
'  Example: WindowsDefaultsDisplayed property
'  Return the default Windows color for text and background in a message box.
    Dim arewindowsdefaultsdisplayed As Variant
    arewindowsdefaultsdisplayed = [].WindowsDefaultsDisplayed
    MessageBox("Window defaults are currently " +arewindowsdefaultsdisplayed + ".")
```

1-2-3: Windows property

{button ,AL(`H_123_APPLICATION_CLASS',0)} See list of classes

(Read-only) Returns a collection of all the open windows in 1-2-3.

Data type

DocWindows

Syntax

Set *docwindows* **=** *application*.**Windows**

Legal values

The value of the Windows property is the collection of all the open windows in 1-2-3.

```
' Example: Windows property
' Return a collection of the open windows.
    Dim windowcollection As DocWindows
    Set windowcollection = CurrentApplication.Windows
```

1-2-3: ZoomScale property

{button ,AL(`H_123_APPLICATION_CLASS;H_123_DOCUMENT_CLASS',0)} See list of classes

{button ,AL(`H_123_ZOOM_PROPERTY_EXSCRIPT;H_123_GRIDLINECOLOR_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the percentage view scale factor for all sheets in the file or application.

Data type

Long

Syntax

object.**ZoomScale =** *value*

value **=** *object*.**ZoomScale**

Legal values

The value of the ZoomScale property is any long from 25 - 400.

1-2-3: Zoom property

{button ,AL(`;H_123_PLOT_CLASS;H_123_SHEET_CLASS',0)} See list of classes

{button ,AL(`H_123_ZOOM_PROPERTY_EXSCRIPT',1)} See example

Sets or returns the percentage view scale factor for the specified document or map plot.

Data type

For documents: Long

For map plots: Double

Syntax

object.**Zoom =** *value*

value **=** *object*.**Zoom**

Legal values

The value of the Zoom property for documents is any percentage view scale factor from 25 - 400.

```
' Example: Zoom and ZoomScale properties; ZoomTo method
'This example illustrates different ways to set the zoom scale.
Sub zooming
   ' First determine the current zoom scale.
   Dim firstzoom As Long
   firstzoom = CurrentDocument.Zoom

   ' Use ZoomTo to set the zoom scale to a standard value: 25, 50, 75, 100, 200.
   ' In this case, set the zoom scale to 50%.
   Messagebox "Set the zoom scale to 50%"
   CurrentDocument.ZoomTo 50

   ' Use Zoom to set the zoom scale to any value between 25 and 400.
   ' In this case, set the zoom scale to 125%.
   Messagebox "Set the zoom scale to 125%"
   CurrentDocument.Zoom = 125

   ' Use ZoomScale to set a new custom value.
   ' In this case, set the custom value to 75%.
   Messagebox "Set the zoom scale to 75%"
   CurrentDocument.ZoomScale = 75

   ' Now set the zoom scale back to its initial value.
   Messagebox "Set the zoom scale back to its initial value (" + Str(firstzoom) + "%)"
   CurrentDocument.Zoom = firstzoom
End Sub
```

1-2-3 LotusScript Classes

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

Application
ApplicationWindow
ApproachConnection
Arc

B

Background
BaseCollection
BaseObject
ButtonControl

C

Chart
Charts
ClassInfo
Color
Colors

D

DataLink
DataLinks
DataQuery
DateTime
Document
Documents
DocWindow
DocWindows
DrawCollection
DrawLine
DrawObject
DrawObjects

E

EditText
Ellipse

F

Font
Freehand

G

Group

H

I

J

K

L

Legend

M

Map
MapBin
MapBins
MapPlot
Maps
MapTextEntries
MapTextEntry
MapTitle
Menu
MenuBar

N

O

OLEObject
OLEObjects

P

Picture
Polygon
Polyline
PrintSettings
PrintSettingsCollection

Q

QueryTable
QueryTables

R

1-2-3 LotusScript Events

Calculate
CancelPrint
CellContentsChange
CellValueChange
Click
CloseWindow
Deselected
DisplayInit
DocumentOpened
EndPrint
GetFocus
Initialize
LostFocus
MenuBarReset
MethodInvoked
Moved
NameChange
Opened
OpenWindow
Poll1
Poll2
Poll3
Poll4
PostClose
PostSave
PostSaveAs
PreClose
PreSave
PreSaveAs
PropertySet
Resized
Selected
SheetChange
StartPrint
Terminate
ValueChange

1-2-3 LotusScript Methods

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

B

C

ClosePreview
CollapseColumn
CollapseRow
ColorFromRGB
Connect
CopyFill
CopySelection
CopySQLToClipboard
CopyToClipboard
CreateComputedField
CreateRangeName
CreateRangeNameFromLabel
CreateRangeNameTable
CreateTable
Cut
CutSelection

D

DataParse
DataParseGuess
DefineNamedStyle
DeleteColumns
DeleteComputedField
DeleteCurrentVersion
DeleteNamedPrintSettings
DeleteNamedStyle
DeleteQuery
DeleteRangeName
DeleteRecords
DeleteRows
DeleteSheet
DeleteVersion
DeleteVersionGroup
DemoteColumn
DemoteRow
DisableItem
Disconnect
Distribution
DragAndFill

E

EnableItem
EndPoll
ExpandColumn
ExpandRow
ExtendedName
ExtendSelection
ExtendSheetSelectionBack
ExtendSheetSelectionForward

F

FieldAggregateType
FieldAlias
FileAdminLinksRefresh
Find
FitTallest
FitWidest
FitWidestNumber

ModifyNamedStyle
Move
MoveCellPointer
MoveOrigin
MovePoint

N

NewApproachConnection
NewArc
NewArrow
NewButton
NewChart
NewDataLink
NewDocument
NewDocWindow
NewDrawLine
NewEditText
NewEllipse
NewFreehand
NewMap
NewMenu
NewMenuBar
NewNamedPrintSettings
NewObject
NewPicture
NewPolygon
NewPolyline
NewQuery
NewQueryTable
NewRectangle
NewRoundedRectangle
NewSheet
NewVersion
NewVersionGroup
Next
NextSplit

O

Open
OpenDocument
OpenDocumentFromInternet
OpenDocumentFromNotes
OutlineColumnsToLevel
OutlineRowsToLevel

P

PageBack
PageForward
Paste
PointX
PointY
Preview
Print
PrintOut
PrintToFile
PromoteColumn
PromoteRow

Q

QuerySortDefineKey
QuickCopy
QuickMove
Quit

R

RangeCombine
RangeCombineText
RangeExtract
RangeFill
RangeSortDefineKey
RangeValue
RecalcRange
RecenterMap
RedefineNamedPrintSettings
RedrawMap
Refresh
RefreshOutput
RefreshQuery
Regression
RegressionReset
Remove
RemoveAllVersions
RemoveFromSelection
RemoveItem
RemoveOverlay
RemoveSelectField
RemoveVersion
RenameNamedPrintSettings
RenameNamedStyle
Replace
ReplaceAll
ReplaceItem
ReplaceMenu
ReportVersion
ReservationGet
ReservationReleased
ResetColumnWidth
ResetFieldAggregates
ResetMenuBar
ResetRowHeight
ResetViewOverrides
Reshape
Resize
Restore
RestoreToOriginalSize
RetrieveFileFromInternet
RetrievePrintSettings
RevertToNamedStyle
RevertToStyle

S

SameColor
Save
SaveAs
SaveAsToInternet
SaveAsToNotes
SaveCopyAs

ScrollToActiveCell
Select
SelectAll
SelectAllSheets
Send
SendCommand
SendMail
SendSQL
SetActiveCell
SetCellData
SetGalleryStyle
SetHorizontalTitle
SetInternetOptions
SetLinkSource
SetOrigin
SetRecordsLimitMax
SetVerticalTitle
Show
ShowAllSheets
ShowIconBar
ShowSheet
SmartSort
SmartSum
Sort
SortData
SortReset
SortResetKeys
StartPoll
StyleFontReset

T

Tile
TileHorizontal
TileVertical
TimeDifference
ToBack
ToFront
ToggleVersionBorder
TopLeftVisibleCell
Transpose
TurnTo

U

UncheckItem
UnGroup
UnGroupSheets
UnhideColumns
UnhideRows
UnloadAddin
Update
UpdateDefaultPrintSettings
UseDefaultPrintSettings
UserLogin

V

Verb
VersionGroup
VersionGroups

1-2-3 LotusScript Properties

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

- Active
- ActiveCell
- ActiveDocument
- ActiveDocWindow
- Addins
- AlignOverColumns
- AllFields
- AllNames
- AllowsUpdates
- AllPagesPrint
- AlwaysReserve
- Anchor
- Application
- ApplicationMaximized
- ApplicationWindow
- ArgumentSeparator
- Arrow
- Author
- Authors
- AutoExecMacrosEnabled
- AutoOpenPath
- AutoRedraw
- AutoRefresh
- AutoUpdate
- AvailableMemory

B

- BackColor
- Background
- BaseMapName

C

IsLocked
IsNew
IsNotesFX
IsParenthesized
IsProtected
IsRangeNamed
IsRowCollapsed
IsRowHidden
IsSelectable
IsSelected
IsSheetHidden
IsZeroDisplayed
Italic
ItemName

J

K

Keywords
KnownRegionAliases
KnownRegionCodes
KnownRegionNames

L

L123Seconds
LabelRange
LabelSource
Language
LastEditor
LastPrinted
LastVersionGroup
Left
LeftBorder
LeftMargin
Legend
Lines
LinkedToCell
LinkSource
LocalTime
LongPrompt
LSLocalTime

M

MacroStep
MacroTrace
MaintainDimensions
Maps
MatchAccent
MatchCase
MatchKatakana
MatchPitch
MenuPrompt
MenuText
Methods
ModifiedDate
MonthNames
MorningString

N

Name
NamedPrintSettings
NamedRanges
NegativesInColor
Normal
NotesPath
NumberOfMostRecentFiles

O

Object
OLEObjects
Orientation
OSType
OutlineBorder
OutputLocation
OutputRange
Overlays

P

PaperBinName
PaperBinNames
PaperHeight
PaperHeightMaximum
PaperHeightMinimum
PaperSizeCustom
PaperSizeName
PaperSizeNames
PaperWidth
PaperWidthMaximum
PaperWidthMinimum
Parent
Password
Path
Pattern
PatternBins
PatternVisible
Placement
Plot
PlotPosition
PlotRotation
PointCount
PrinterName
PrinterNames
PrinterQuality
PrintPagesFrom
PrintPagesStart
PrintPagesTo
PrintRange
PrintRangeSaved
PrintSelection
PrintWhat
ProductVersion
Properties

Q

QueryTables

R

S

V

Value
ValueRange
ValueSource
VersionBorderVisible
VersionId
VersionName
VersionStatus
VerticalBorder
VerticalPageBreak
VerticalScrollBarVisible
VerticalTitle
ViewSplitHeight
ViewSplitStyle
ViewSplitWidth
Visible

W

WelcomeOn
WideUnderline
Width
Windows
WindowsDefaultsDisplayed

X

Y

Z

Zoom
ZoomScale

1-2-3 LotusScript A-Z

A

B

C

D

E

F

G

H

N

O

P

Q

R

S

V

W

X

Y

Z