Approach: Application class

The Application maintains application-wide settings and user information. There can be only one application object per running application, that is, one application per Approach executable running. There is a single window associated with each application.

Contained by

Class
None

Usage

Use the Application class to find information about Approach, such as the following:

- If the application is active.
- The application's name and path.
- The active view.
- The icon sets defined and current icon sets.
- The menus defined and current menu bar.

Application class members

Properties

- ActiveDocument
- ActiveDocWindow
- ActiveView
- Application
- ApplicationWindow
- Documents
- FullName
- IconSets
- Language
- Menus
- Name
- Parent
- Path
- Selection
- Visible
- Windows

Methods

- CloseWindow
- GetColorFromRGB
- OpenDocument
- RunProcedure

Events

- Broadcast
- DocumentClose
- DocumentOpened
- MailCheck
- MailReceived
- MailSend
- Quit

Approach: ApplicationWindow class

The ApplicationWindow describes the main Approach window for a running application. There is one application window for each running Approach executable.

Contained by

Class
Application

Usage

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

Use ApplicationWindow to do the following:

- Determine the status of menu commands, such as whether Show Data is enabled.
- Operate window commands, such as minimize or maximize the window.

ApplicationWindow class members

Properties

- Application
- Parent

Methods

- Cascade
- Close
- DoMenuCommand
- GetHandle
- IsCommandChecked
- IsCommandEnabled
- Maximize
- Minimize
- Restore
- Tile

Events

- None

Approach: Background class

The background style for displayable objects that have Background as a property. Determines the color of an object's background.

Contained by

Class
Checkbox
DropdownBox
Ellipse
Fieldbox
ListBox
Panel
RadioButton
Rectangle
RoundRect
Textbox

Usage

Use this class to modify the backgrounds of objects which have a Background property. You can create and modify background objects. Once you have a background object, you can use it as the property of any object with a Background property. A background object cannot be displayed by itself; it is used as the Background property of another object.

Background class members

Properties

Color

Methods

None

Events

None

Approach: BaseCollection class

BaseCollection provides the basic collection properties and methods to determine the number of objects in the collection and whether or not the collection is empty

Contained by

Class
None

Usage

The BaseCollection class is an abstract class. That is, you cannot create an instance of BaseCollection. You can, however, represent all of BaseCollection's subclasses with the BaseCollection class. For example, you can write a subroutine which takes BaseCollection as a parameter. This allows you to pass any instance of a BaseCollection subclass to that subroutine. You can also use LotusScript functions to iterate through a Collection class.

BaseCollection class members

Properties

Count

Methods

IsEmpty

Events

None

Approach: BodyPanel class

The body of a view shows data from individual records.

Contained by

Class
ChartView
Crosstab
Envelope
Form
FormLetter
MailingLabels
Report
Worksheet

Usage

Use a BodyPanel object to set the display characteristics of the view panel, such as the following:

- Color of the background and border.
- Printing behavior, such as contracting to fit the size of the panel contents.

BodyPanel class members

Properties

- Background
- Border
- Expand
- Height
- NamedStyle
- Parent
- Reduce
- Type

Methods

- MakeNamedStyle

Events

- None

Approach: Border class

The border style for displayable objects.

Contained by

Class
BodyPanel
DropdownBox
Ellipse
FieldBox
HeaderFooterPanel
ListBox
Rectangle
RepeatingPanel
RoundRect
SummaryPanel
TextBox

Usage

Use this class to manipulate the borders of objects that have the Border property. You can create and modify border objects. Once you have a border object, you can use it as the property of any object with a border property.

Border class members

Properties

- BaseLine
- Bottom
- Color
- EncloseLabel
- Left
- Pattern
- Right
- Top
- Width

Methods

- None

Events

- None

Approach: Button class

A button that runs an Approach macro or calls a LotusScript function or procedure when the button is clicked.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to provide quick starts for Approach macros or LotusScript functions or procedures. The macro, function, or procedure can perform one or more actions. For example, you can use a button to run a macro to switch forms.

Button class members

Properties

- Enabled
- Font
- Height
- Left
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Text
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: ChartView class

A ChartView object is an Approach chart.

Contained by

Class
Document

Usage

Use a ChartView object to determine characteristics of a chart view, such as the following:

- The Approach file (document) that the view is in.
- The status of the display of menu bars for the view.
- Macros activated by switching to or from the view.

ChartView class members

Properties

- Document
- MainTable
- MenuBar
- Name
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- TimerInterval
- Type
- Visible

Methods

- New

Events

- SwitchFrom
- SwitchTo
- UserTimer

Approach: CheckBox class

A control indicating a yes/no or a true/false condition. A checkbox can be checked or unchecked.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to record user input. If it is checked, indicating yes or true, an X is displayed in the box. If it is unchecked, indicating no or false, the box is empty.

CheckBox class members

Properties

- Background
- CheckedValue
- DataField
- DataTable
- Height
- IsChecked
- LabelAlignment
- LabelFont
- LabelPosition
- LabelText
- Left
- MacroDataChange
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ReadOnly
- Relief
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- UncheckedValue
- Value
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus
- SetState
- SetValue

Events

- Change
- Click
- DoubleClick
- GotFocus
- LostFocus
- MouseDown
- MouseMove
- MouseUp

Approach: Collection class

A Collection object defines methods to add, delete, and modify lists of objects. A Collection object is a collection of objects.

Contained by

Class
None

Usage

Use a Collection object to make changes to a list of objects. For example, use a Collection to remove or replace display objects in a form.

Collection class members

Properties

- Count

Methods

- Add
- IsEmpty
- Merge
- New
- Remove
- SetAt

Events

- None

Approach: Color class

A Color object defines a color for objects that have Color as a property.

Contained by

Class
Background
Border
Font
LineStyle

Usage

Use the properties of this class to set and read the current levels of red, blue, and green that contribute to a specific color. You cannot change the individual components of a color, but you can modify the color. A color object cannot be displayed; it is used as the Color property of another object.

Color class members

Properties

- Black
- Blue
- Cyan
- Green
- Magenta
- Red
- Transparent
- Yellow

Methods

- GetRGB
- New
- SameColor
- SetRGB

Events

- None

Approach: Connection class

Represents a connection to a database.

Contained by

Class
None

Usage

Use this class to connect to a database. You can also get information on a connection from this class, such as the following:

- The name and path of the database file.
- The user ID of the file's owner.

Connection class members

Properties

- DataSourceName
- IsConnected
- Password
- UserId

Methods

- ConnectTo
- Disconnect
- GetError
- GetErrorMessage
- GetExtendedErrorMessage
- ListDataSources
- ListFields
- ListTables
- New

Events

- None

Approach: Crosstab class

A Crosstab object is an Approach crosstab view.

Contained by

Class
Document

Usage

Use a Crosstab object to determine characteristics of a crosstab view, such as the following:

- The Approach file (document) that the view is in.
- If the crosstab reflects the current found set.
- The drill-down view for the crosstab.

Crosstab class members

Properties

- ApplyFoundSet
- Document
- DrillDownView
- MainTable
- MenuBar
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- PrintDate
- PrintPageNum
- PrintTitle
- ShowRelated
- TimerInterval
- Title
- Type
- Visible

Methods

- New

Events

- SwitchFrom
- SwitchTo
- UserTimer

Approach: Display class

The display class is a base class to graphically represented classes like button, checkbox, picture, and so on.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

The Display class is an abstract class. That is, you cannot create an instance of Display. You can, however, represent all of the subclasses of display with the Display class. For example, you can write a subroutine which takes Display as a parameter. This allows you to pass any instance of a Display subclass to that subroutine.

The coordinate system for display objects is based on the top left corner of the panel in which the object is placed. For example, if you have a picture object in the header of a report, its coordinates are based on the top left corner of the header panel, not the report. Similarly, for a checkbox placed in the body of the report, its coordinates are based on the top left corner of the body panel, not the report.

Display class members

Properties

- Height
- Left
- MacroTabIn
- MacroTabOut
- Name
- NonPrinting
- Page
- Parent
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: Document class

A Document is a class which describes the basic top-level Approach file (.APR) for an application.

Contained by

Class
Application

Usage

Create a Document for each Approach file.

A document contains information such as the following:

- File name and path of the Approach file.
- The owner or author's name.
- A description of the document.
- Creation and modification dates for the Approach file.
- The databases associated with the Approach file.
- The macros and named find/sorts associated with the document.

Document class members

Properties

- Author
- CalcTable
- CreateDate
- Description
- FileName
- FullName
- Keywords
- LastModified
- Menus
- Modified
- Name
- NamedFindSorts
- NamedStyles
- NumJoins
- NumRevisions
- NumTables
- NumViews
- Parent
- Path
- Tables
- User
- VarTable
- Views
- Window

Methods

- Activate
- GetTableByName
- New

Events

- None

Approach: DocWindow class

A DocWindow object provides access to the current state of the document, including such information as the current record and view.

Contained by

Class
Document

Usage

Use a DocWindow object to use views to perform database operations, such as the following:

- Navigate between fields in a view.
- Navigate between records in the current found set.
- Enter data into fields.

DocWindow class members

Properties

- ActionBarVisible
- ActiveView
- CurrentRecord
- Document
- IconBarVisible
- NamedFindSorts
- NumRecordsFound
- Parent
- Redraw
- StatusBarVisible
- ViewTabVisible
- Visible

Methods

- Close
- FillField
- FirstRecord
- GetHandle
- LastRecord
- Maximize
- Minimize
- NextRecord
- PrevRecord
- Repaint
- Restore
- TabNext
- TabPrev

Events

- CloseWindow
- NewRecord
- OpenWindow
- RecordChange
- RecordCommit
- ViewSwitch

Approach: DropDownBox class

A DropDownBox object displays a list of values in a drop-down box. The user can select only one item in the list.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to display lists of possible selections for a user. For example, a drop-down box can display the existing values in a field to simplify data entry.

DropDownBox members

Properties

- Alignment
- Background
- Border
- Editable
- Font
- Height
- LabelAlignment
- LabelFont
- LabelPosition
- LabelText
- Left
- MacroDataChange
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ReadOnly
- ShadowColor
- ShowArrow
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Text
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFieldList
- SetFocus
- SetList

Events

- Change
- Click
- DoubleClick
- GotFocus
- LostFocus
- MouseDown
- MouseUp

Approach: Ellipse class

An Ellipse object describes a drawn object on a panel in the shape of an ellipse. An ellipse with its height equal to its width is a circle.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use an Ellipse object to describe the following characteristics of an ellipse:

- The position of the object, including which page of a view.
- The background and border colors.
- The object's printing behavior.
- The object's place in the data-entry tab order.

Ellipse class members

Properties

- Background
- Border
- Height
- Left
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: Envelope class

An Envelope object is an Approach envelope view.

Contained by

Class
Document

Usage

Use an Envelope object to determine characteristics of an envelope view, such as the following:

- The current selection, if any.
- The Approach file (document) that the view is in.
- The collection of objects contained in the view.

Envelope class members

Properties

- Document
- HideMargins
- MainTable
- MenuBar
- Name
- ObjectList
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- Selection
- TimerInterval
- Type
- Visible

Methods

- None

Events

- SwitchFrom
- SwitchTo
- UserTimer

Approach: FieldBox class

A control that allows for data entry and displays data. The data may or may not be from a field in a database.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to allow the user to enter data. For example, a mailing label might use an address field where the address can be entered and later displayed. The address data could also come from a customer database where one of the pieces of information in the database is the customer's address. This example is a "bound" field box.

An "unbound" field box could be used to provide a value for a calculation. For example, a report might use a field box for the user to enter a value that is used to determine which records are summarized in the report.

FieldBox class members

Properties

- Alignment
- Background
- Border
- DataField
- DataTable
- Expand
- Font
- Height
- LabelAlignment
- LabelFont
- LabelPosition
- LabelText
- Left
- MacroDataChange
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Parent
- ReadOnly
- Reduce
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Text
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Change
- Click
- DoubleClick
- GotFocus
- KeyDown
- KeyPress
- KeyUp
- LostFocus
- MouseDown
- MouseUp

Approach: Font class

A text style for objects that have text and have Font as a property. You can modify the color, size, and name of the font.

Contained by

Class
Button
DropDownBox
FieldBox
ListBox
TextBox

Usage

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

Font class members

Properties

- Bold
- Color
- FontName
- Italic
- Relief
- Size
- Strikethrough
- Underline

Methods

- None

Events

- None

Approach: Form class

A Form object is an Approach form.

Contained by

Class
Document

Usage

Use a Form object to determine characteristics of a form view, such as the following:

- What object is currently selected, if any.
- The Approach file (document) that the view is in.
- The collection of objects contained in the view.
- The number of pages in the view.

Form class members

Properties

- CurrentPageNum
- Document
- HideMargins
- MainTable
- MenuBar
- Name
- NumPages
- ObjectList
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- Selection
- ShowAsDialog
- TimerInterval
- Type
- Visible

Methods

- DeletePage
- New
- NewPage

Events

- SwitchTo
- UserTimer

Approach: FormLetter class

A FormLetter object is an Approach form letter.

Contained by

Class
Document

Usage

Use a FormLetter object to determine characteristics of a form letter view, such as the following:

- What object is currently selected, if any.
- The Approach file (document) that the view is in.
- The collection of objects contained in the view.

FormLetter class members

Properties

- CurrentPageNum
- Document
- HideMargins
- MainTable
- MenuBar
- Name
- NumPages
- ObjectList
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- Selection
- TimerInterval
- Type
- Visible

Methods

- None

Events

- SwitchFrom
- SwitchTo
- UserTimer

Approach: HeaderFooterPanel class

A HeaderFooterPanel object describes the header or footer information that appears in an Approach report.

Contained by

Class
Report

Usage

Use the HeaderFooterPanel object to control characteristics of headers and footers, such as the following:

- The Named Style used to display the panel.
- The color of the background and border of the panel.
- The size of the panel.

HeaderFooterPanel class members

Properties

- Background
- Border
- Height
- NamedStyle
- Parent
- Type

Methods

- MakeNamedStyle

Events

- None

Approach: LineObject class

Displays a horizontal, vertical, or diagonal line on a form, report, mailing label, form letter, envelope, or chart.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use a LineObject object to add lines to an Approach view. For example, use a LineObject to divide sections of the body of a form to clarify field data groupings.

You can control the characteristics of the LineObject, such as the following:

- The Named Style used to display the object.
- The color of the object and its shadow.
- The width and height of the object.
- The position of the object, including the view it appears on.
- The printing behavior of the object.

LineObject class members

Properties

- Background
- Height
- Left
- LineStyle
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Orientation
- Page
- Parent
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: LineStyle class

A line style for objects that have LineStyle as a property. You can create and modify LineStyles. A LineStyle object itself cannot be displayed, it must be used as the LineStyle property of an object.

Contained by

Class
LineObject

Usage

Use this class to modify lines styles used for display objects. A LineStyle object cannot be displayed; it is used as the LineStyle property of another object.

LineStyle class members

Properties

- Color
- Pattern
- Width

Methods

- None

Events

- None

Approach: ListBox class

A control that displays a list of values. A ListBox object is a Field box & list in an Approach view.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to display lists of specific information. Use a list box when you want to limit the values the user can select to the values in the list.

ListBox class members

Properties

- Background
- Border
- Font
- Height
- LabelAlignment
- LabelFont
- LabelPosition
- LabelText
- Left
- MacroDataChange
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ReadOnly
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Text
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFieldList
- SetFocus
- SetList

Events

- Change
- Click
- DoubleClick
- GotFocus
- LostFocus
- MouseDown
- MouseMove
- MouseUp

Approach: MailingLabels class

A MailingLabels object is an Approach mailing label view.

Contained by

Class
Document

Usage

Use a MailingLabels object to determine characteristics of a mailing label view, such as the following:

- What object is currently selected, if any.
- The Approach file (document) that the view is in.
- The collection of objects contained in the view.

MailingLabels class members

Properties

- Document
- HideMargins
- MainTable
- MenuBar
- Name
- ObjectList
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- Selection
- TimerInterval
- Type
- Visible

Methods

- None

Events

- SwitchFrom
- SwitchTo
- UserTimer

Approach: OLEObject class

An OLE object from any OLE-enabled application.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to bring in objects from other applications. OLE objects can be anything, such as a document from a word processor or a spreadsheet from a spreadsheet program. An OLE object can be an entire file, such as a spreadsheet, or part of a file, such as a chart from a spreadsheet.

OLEObject class members

Properties

- Dispatch
- Height
- Left
- LPObject
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NonPrinting
- Page
- Parent
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- DoVerb
- InsertAfter
- MakeNamedStyle
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseMove
- MouseUp

Approach: Panel class

The Panel class is a abstract class to classes such as BodyPanel, RepeatingPanel, HeaderFooterPanel, and SummaryPanel.

Contained by

Class
ChartView
Crosstab
Envelope
Form
FormLetter
MailingLabels
Report
Worksheet

Usage

The Panel class is an abstract class. That is, you cannot create an instance of a Panel. You can, however, represent all of Panel's subclasses with the Panel class. For example, you can write a subroutine which takes Panel as a parameter. This allows you to pass any instance of a Panel subclass to that subroutine.

Panel class members

Properties

- Background
- Border
- Height
- NamedStyle
- Parent
- Type

Methods

- MakeNamedStyle

Events

- None

Approach: Picture class

An image you can display on a form, report, mailing label, form letter, envelope, or chart. The picture appears for every record or page of records in the view.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to display images. Pictures can be used to visually enhance views of data. Use PicturePlus class to store images as part of records.

Picture class members

Properties

- Background
- Height
- Left
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus
- SetPicture

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: PicturePlus class

A control that displays an image or any OLE object stored in an Approach database.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to display images and OLE objects. You can also create or edit the images or OLE objects.

PicturePlus class members

Properties

- AllowDrawing
- Background
- Border
- DataField
- DataTable
- Display
- Height
- Left
- MacroDataChange
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- Position
- ReadOnly
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- Stretch
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: Query class

An SQL statement or TableName which queries a connection.

Contained by

Class
None

Usage

Use this class to define the data you want to retrieve.

Query class members

Properties

- Connection
- SQL
- TableName

Methods

- GetError
- GetErrorMessage
- GetExtendedErrorMessage
- New

Events

- None

Approach: RadioButton class

A RadioButton object is a data-entry control indicating a mutually exclusive selection.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Use this class to allow users to enter a selection when you want only one selection at a time. A RadioButton can be selected or not. If it is selected, the center is black. Typically, radio buttons come in groups. When one RadioButton in the group is selected, the others become deselected.

Note You cannot create radio button groups from Script, but you can set or unset the attribute of a radio button that is part of a group from Script.

RadioButton class members

Properties

- Background
- ClickedValue
- DataField
- DataTable
- Height
- IsClicked
- LabelAlignment
- LabelFont
- LabelPosition
- LabelText
- Left
- MacroDataChange
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ReadOnly
- Relief
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Value (RadioButton)
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus
- SetValue

Events

- Change
- Click
- DoubleClick
- GotFocus
- LostFocus
- MouseDown
- MouseUp

Approach: Rectangle class

Displays a rectangle on a form, report, mailing label, form letter, envelope, or chart.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Rectangles can be used to visually enhance views of data. For example, use a rectangle to group fields in a form.

Rectangle class members

Properties

- Background
- Border
- Height
- Left
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: RepeatingPanel class

A RepeatingPanel object shows data from more than one record from a joined database that correspond to the current record in a form. For example, use this class to create a panel to show the relationship between a department and the employees in the department.

Contained by

Class
Form

Usage

Use a repeating panel to show a one-to-many relationship between the records in two databases. Both databases must be joined in the Approach file (document).

The RepeatingPanel class controls the following characteristics of a repeating panel:

- Color of the background and border.
- Printing behavior, such as contracting to fit the size of the panel contents.
- The position of the panel on the form.

RepeatingPanel class members

Properties

- AlternateColors
- Background
- Border
- Height
- Left
- MainTable
- Name
- NamedStyle
- NumLines
- Page
- Parent
- Top
- Type
- Width

Methods

- MakeNamedStyle
- New

Events

- None

Approach: Report class

A Report object is an Approach report view.

Contained by

Class
Document

Usage

Use a Report object to determine characteristics of a report view, such as the following:

- What object is currently selected, if any.
- The Approach file (document) that the view is in.
- The collection of objects contained in the view.
- The number of columns that data appears in on the report.

Report class members

Properties

- Document
- KeepRecsTogether
- MainTable
- MenuBar
- Name
- NumColumns
- ObjectList
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- TimerInterval
- Type
- Visible

Methods

- New

Events

- SwitchFrom
- SwitchTo
- UserTimer

Approach: ResultSet class

A set of records retrieved from a query. Data in a ResultSet can be edited.

Contianed by

Class
None

Usage

Use a ResultSet object to work with a found set of data. You can perform operations with the data such as the following:

- Determine the field size, name, and type.
- Navigate between fields, rows, or columns in a record or group of records.
- Edit values in fields.

ResultSet class members

Properties

- CurrentRow
- IsBeginOfData
- IsEndOfData
- IsReadOnly
- IsResultSetAvailable
- Query

Methods

- AddRow
- Close (ResultSet)
- DeleteRow
- Execute
- FieldExpectedDataType
- FieldID
- FieldName
- FieldNativeDataType
- FieldSize
- FirstRow
- GetError
- GetErrorMessage
- GetExtendedErrorMessage
- GetParameter
- GetParameterName
- GetValue
- LastRow
- New
- NextRow
- NumColumns
- NumParameters
- NumRows
- Options
- PreviousRow
- SetParameter
- SetValue
- UpdateRow

Events

- None

Approach: RoundRect class

Displays a round rectangle on a form, report, mailing label, form letter, envelope, or chart.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Round rectangles can be used to visually enhance views of data. For example, use a round rectangle to separate sections fields in a form.

RoundRect class members

Properties

- Background
- Border
- Height
- Left
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: SummaryPanel class

A SummaryPanel object displays fields or calculations for groups of records.

Contained by

Class
ChartView
Report

Usage

Use a SummaryPanel object to show calculations involving more than one record or to distinguish record groupings.

Use a Body object to set the display characteristics of the view panel, such as the following:

- Color of the background and border.
- Printing behavior, such as contracting to fit the size of the panel contents.
- The records used in the group represented by the panel.

SummaryPanel class members

Properties

- Alignment
- Background
- Border
- Expand
- GroupByDataField
- GroupByDataTable
- GroupByEvery
- Height
- Location
- Name
- NamedStyle
- PageBreak
- Parent
- Reduce
- Type
- Width

Methods

- MakeNamedStyle
- New

Events

- None

Approach: Table class

The Table object represents the information of a database accessed through Approach.

Contianed by

Class
Document

Usage

Use a Table object to retrieve information about the database, such as the following:

- The name and path of the database.
- The database field names, sizes, and types.
- The number of records in the database.

Table class members

Properties

- FieldNames
- FileName
- FullName
- NumFields
- NumRecords
- Parent
- Path
- TableName

Methods

- CreateResultSet
- GetFieldFormula
- GetFieldOptions
- GetFieldSize
- GetFieldType
- ReplaceWithResultSet

Events

- None

Approach: TextBox class

A TextBox displays text on an Approach view.

Contained by

Class
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel

Usage

Text boxes can be used to visually enhance views or provide information to users. For example, use a text box to provide instructions for making selections from check boxes or radio buttons on a form.

Use a TextBox object to specify the characteristics of the text box, such as the following:

- The named style used to determine the display of the object.
- The position of the object, including which page of a view.
- The background and border colors.
- The object's printing behavior.
- The object's place in the data-entry tab order.

TextBox class members

Properties

- Alignment
- Background
- Border
- Font
- Height
- Left
- LineSpacing
- MacroClick
- MacroTabIn
- MacroTabOut
- Name
- NamedStyle
- NonPrinting
- Page
- Parent
- ShadowColor
- ShowInPreview
- SlideLeft
- SlideUp
- TabOrder
- TabStop
- Text
- Top
- Type
- Visible
- Width

Methods

- BringToFront
- InsertAfter
- MakeNamedStyle
- New
- Refresh
- SendToBack
- SetFocus

Events

- Click
- DoubleClick
- MouseDown
- MouseUp

Approach: View class

The view class is a base class for the creation of view objects, such as forms, form letters, reports, and worksheets.

Contained by

Class
Document

Usage

The View class is an abstract class. That is, you cannot create an instance of View. You can, however, represent all of View's subclasses with the View class. For example, you can write a subroutine which takes View as a parameter. This allows you to pass any instance of a View subclass to that subroutine.

For example, you would have a subroutine designed to adjust margins. You don't know what type of view will be passed to the subroutine so you use View as a parameter. This way you can pass any View subclass to the subroutine and it will run.

View class members

Properties

- Document
- MainTable
- MenuBar
- Name
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- TimerInterval
- Type
- Visible

Methods

- None

Events

- SwitchFrom
- SwitchTo
- UserTimer

Approach: Window class

The window class is a base class for the creation of objects such as the ApplicationWindow and DocWindow.

Contained by

Class
Application
Document

Usage

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

Window class members

Properties

None

Methods

Close
GetHandle
Maximize
Minimize
Restore

Events

None

Approach: Worksheet class

A Worksheet object is an Approach worksheet view.

Contained by

Class
Document

Usage

Use a Worksheet object to determine characteristics of a worksheet view, such as the following:

- What object is currently selected, if any.
- The Approach file (document) that the view is in.
- The collection of objects contained in the view.

Worksheet class members

Properties

- Document
- MainTable
- MenuBar
- Name
- OnSwitchFromMacro
- OnSwitchToMacro
- Parent
- PrintDate
- PrintPageNum
- PrintTitle
- TimerInterval
- Title
- Type
- Visible

Methods

- AddColumn
- GetText
- New
- RemoveColumn
- SetCellFocus
- SetText

Events

- CellDataChange
- CellGetFocus
- CellLostFocus
- SelectColumn
- SwitchFrom
- SwitchTo
- UserTimer

Approach: ActionBarVisible property

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

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

Hides or displays the action bar at the top of the application window.

Data type

Integer

Syntax

docwintitle.**ActionBarVisible** = *flag*

flag = *docwintitle*.**ActionBarVisible**

Legal values

Value	Description
TRUE	(Default) Display the action bar.
FALSE	Do not display the action bar.

Approach: ActiveDocument property

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

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

(read-only) Returns the name of the currently active document.

Data type

Document

Syntax

doc = *appname*.**ActiveDocument**

Legal values

The default value for the ActiveDocument property is none, or the current active document.

Approach: ActiveDocWindow property

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

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

Sets or returns the currently active document window.

Data type

DocWindow

Syntax

appname.**ActiveDocWindow** = *docwindow*

docwindow = *appname*.**ActiveDocWindow**

Legal values

The default value for the ActiveDocWindow property is none, or the current active document window.

Approach: ActiveView property

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

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

Sets or returns the currently active view in the document.

Data type

Variant

Syntax

appname.docname.**ActiveView** = *view*

view = *appname.docname*.**ActiveView**

or

docwindowname.**ActiveView** = *view*

view = *docwindowname*.**ActiveView**

Legal values

The default value for the ActiveDocument property is none, or the current active view.

Approach: Alignment property

{button ,AL(`H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Aligns the object to the left, right, or center of the document on which the object is placed.

Data type

Long

Syntax

objectname.**Alignment** = *value*

value = *objectname*.**Alignment**

Legal values

Value	Description
$ItsAlignmentLeft	(Default) Align the object to the left.
$ItsAlignmentHorizCenter	Align the object in the center.
$ItsAlignmentRight	Align the object to the right.

Approach: AllowDrawing property

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

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

Sets or determines if you can draw with the mouse in PicturePlus objects.

Data type

Integer

Syntax

pictureplusobject.**AllowDrawing** = *flag*

flag = *pictureplusobject*.**AllowDrawing**

Legal values

Value	Description
TRUE	(Default) Users can draw in the PicturePlus object.
FALSE	Users cannot draw in the PicturePlus object.

Approach: AlternateColors property

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

Alternates the line colors of the records in a repeating panel. The alternating colors used are the background color of the panel and the background color of the form.

Data type

Integer

Syntax

repeatingpanelobject.**AlternateColors** = *flag*

flag = *repeatingpanelobject*.**AlternateColors**

Legal values

Value	Description
TRUE	(Default) Display alternate colors.
FALSE	Do not display alternate colors.

Approach: Application property (ApplicationWindow class)

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

(read-only) Returns the application object to which the window belongs.

Data type

Application

Syntax

application = *applicationwindow*.**Application**

Approach: Application property (Application class)

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

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

(read-only) Points to the application itself.

Data type

Application

Syntax

appname = *applicationwindow*.**Application**

Approach: ApplicationWindow property

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

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

(read-only) Returns the application window.

Data type

ApplicationWindow

Syntax

applicationwindow = *application*.**ApplicationWindow**

Legal values

The default for the ApplicationWindow property is the application window.

Approach: ApplyFoundSet property

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

Applies the current found set to a crosstab.

Data type

Integer

Syntax

crosstab.**ApplyFoundSet** = *flag*

flag = *crosstab*.**ApplyFoundSet**

Legal values

Value	Description
TRUE	(Default) Apply the found set to a crosstab.
FALSE	Do not apply the found set to a crosstab.

Approach: Author property

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

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

(read-only) Returns the name of the person who created the document, or the name of the person who last entered their name as the author.

Data type

String

Syntax

name = *docname*.**Author**

Legal values

You can set this property to any string up to N characters. The default value for the Author property is the name of the person who created the document.

Approach: Background property

{button ,AL(`H_BODYPANEL_CLASS;H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_HEADERFOOTERPANEL_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_PANEL_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the background object.

Data type

Background

Syntax

set objectname.**Background** = *backgroundobject*

set backgroundobject = *objectname*.**Background**

Legal Values

You can set this property to any background object.

Approach: Baseline property

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

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

Displays a dashed line in an object just above the bottom border line. For example, a baseline in a field box illustrates where text can be written.

Data type

Integer

Syntax

objectname.Border**.Baseline** = *flag*

flag = *objectname*.Border**.Baseline**

or

*borderobject***.Baseline** = *flag*

flag = *borderobject***.Baseline**

Legal values

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

Approach: Black property

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

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

(read-only) Returns the black component of an object.

Data type

Integer

Syntax

value = *colorobjectname*.**Black**

Legal values

Any integer from 0 to 255.

Usage

0 represents no black and 255 represents the highest amount of black available.

Approach: Blue property

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

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

(read-only) Returns the blue component of an RGB value for an object.

Data type

Integer

Syntax

value = *colorobjectname*.**Blue**

Legal values

Any integer from 0 to 255.

Usage

0 represents no blue and 255 represents the highest amount of blue available. Use the SetRGB method to change the settings of the Red, Green, and Blue properties at once.

Approach: Bold property

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

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

Sets or returns whether the text is bold type.

Data type

Integer

Syntax

objectname.Font**.Bold** = *flag*

flag = *objectname*.Font**.Bold**

or

*fontobject***.Bold** = *flag*

flag = *fontobject***.Bold**

Legal values

Value	Description
FALSE	(Default) The font is not bold.
TRUE	The font is bold.

Approach: Border property

{button ,AL(`H_BODYPANEL_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_HEADERFOOTERPANEL_CLASS;H_LISTBOX_CLASS;H_PANEL_CLASS;H_PICTUREPLUS_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the border object.

Data type

Border

Syntax

set objectname.**Border** = *borderobject*

set borderobject= objectname.**Border**

Legal Values

You can set this property to any border object.

Approach: Bottom property

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

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

Sets or returns whether the bottom border of an object is displayed.

Data type

Integer

Syntax

objectname.Border**.Bottom** = *flag*

flag = *objectname*.Border**.Bottom**

or

*borderobject***.Bottom** = *flag*

flag = *borderobject***.Bottom**

Legal values

Value	Description
TRUE	(Default) Display the bottom border.
FALSE	Do not display the bottom border.

Approach: CheckedValue property

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

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

Sets or returns the value of a check box if it is checked.

Data type

String

Syntax

checkboxobject.**CheckedValue** = *value*

value = *checkboxobject*.**CheckedValue**

Legal values

You can set this property to any string.

Approach: ClickedValue property

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

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

Sets or returns the value of a radio button if it is clicked.

Data type

String

Syntax

radiobuttonobject.**ClickedValue** = *value*

value = *radiobuttonobject*.**ClickedValue**

Legal values

You can set this property to any string.

Approach: Color property

{button ,AL(`H_BACKGROUND_CLASS;H_BORDER_CLASS;H_FONT_CLASS;H_LINESTYLE_CLASS;',0)} See list of classes

Specifies the color object for the object.

Data type

Color

Syntax

set objectname.**Color** = *colorobject*

set colorobject = *objectname*.**Color**

Legal values

You can set this property to any color object. The default color for the Color property is white.

Approach: Connection property

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

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

Specifies the Connection object used for the query.

Data type

Connection

Syntax

*set qreryobject.***Conection** = *connectionobject*

set connectionobject = *qreryobject*.**Connection**

Legal values

You can set this property to any connection object.

Approach: Count property

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

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

(read-only) Returns the number of objects in the collection.

Data type

Integer

Syntax

value = *collectionobject*.**Count**

Legal values

The default value for the Count property is the number of objects in the collection.

Approach: CurrentPageNum property

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

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

Sets or returns the current page number of a form, or form letter.

Data type

Integer

Syntax

formname.**CurrentPageNum** = *value*

value = *formname*.**CurrentPageNum**

Legal values

You can set the property to any number between 1 and the number of pages you have on a form. The default value for the CurrentPageNum property is the current page number of the form or form letter.

Approach: CurrentRow property

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

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

Sets or returns the current row in a found set. Setting the current row value is equivalent to positioning to that row and may require fetching records. The current row number reported by the object is always valid. A zero indicates that no result set is available.

Data type

Long

Syntax

resultsetname.**CurrentRow** = *value*

value = *resultsetname*.**CurrentRow**

Legal values

You can set the property to any number between 1 and the number of rows in the found set. There is no default value for the CurrentRow property.

Approach: Cyan property

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

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

(read-only) Returns the cyan component of a CMYK value for an object.

Data type

Integer

Syntax

value = *colorobjectname*.**CYAN**

Legal values

Any integer from 0 to 255.

Usage

0 represents no cyan and 255 represents the highest amount of cyan available.

Approach: DataField property

{button ,AL(`H_CHECKBOX_CLASS;H_FIELDBOX_CLASS;H_PICTUREPLUS_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the field bound to the display object, such as a field box or PicturePlus object.

Data type

String

Syntax

objectname.**DataField** = *fieldname*

fieldname = *objectname*.**DataField**

Legal values

You can set the property to any field in the database.

Approach: DataSourceName property

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

(read-only) Returns the name describing the type of data.

Data type

String

Syntax

source = connectionobjectname.**DataSourceName**

Legal values

Value	The database is
Lotus Notes - Local	A Lotus Notes local
Lotus Notes - Server	A Lotus Notes Server
Lotus Notes - Workspace	A Lotus Notes Workspace
Access 1.x	Microsoft Access v1 (.MDB)
ODBC	An ODBC file, read-only. (For ODBC you must specify the other ODBC parameters example (DSN;UID;PASSWORD)
Delemited Text	A Delimited Text file, read-only
dBASE III+	dBASE III+ (.DBF)
dBASE IV	dBASE IV (.DBF)
IBM DB2	IBM (.DB2)
Foxpro	Foxpro (.DBF)
Query	Approach Query (.QRY)
Paradox	Paradox 3.x, 4.x (.DB)
SQL Server	SQL Server ()
DB2 - MDI	DB2 via MDI Gateway
Oracle	Oracle ()
Lotus 1-2-3	Lotus 1-2-3 (.WK*)
Excel	Excel (.XLS)
Fixed Length Text	Fixed Length Text file
Access 2.x	Microsoft Access v2, read-only (.MDB)

Approach: DataTable property

{button ,AL(`H_CHECKBOX_CLASS;H_FIELDBOX_CLASS;H_PICTUREPLUS_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the name of the table bound to the display object, such as a field box or PicturePlus object.

Data type

String

Syntax

objectname.**DataTable** = *tablename*

tablename = *objectname*.**DataTable**

Legal values

You can set the property to any of the open tables in the document.

Approach: Document property

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_DOCWINDOW_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABEL_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

(read-only) Returns the document object to which the current window belongs.

Data type

Document

Syntax

set document = docwindow.**Document**

Approach: Documents property

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

(read-only) A collection of open documents in the application.

Data type

BaseCollection

Syntax

set colldocs = applicationobjectname.**Documents**

Legal values

N/A

Approach: DrillDownView property

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

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

Sets or returns the drill-down view of a crosstab.

Data type

Variant

Syntax

crosstabview.**DrillDownView** = *viewname*

viewname = *crosstabview*.**DrillDownView**

Legal values

You can set the property to any view in the document. To set a view as drill-down, pass a view object reference. To unset the property, pass zero.

Approach: Editable property

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

Sets or returns whether you can edit text in a dropdown box.

Data type

Integer

Syntax

ddboxname.**Editable** = *flag*

flag = *ddboxname*.**Editable**

Legal values

Value	Description
TRUE	(Default) You can edit text in the drop-down box.
FALSE	You cannot edit text in the drop-down box, but you can select an item from the list.

Approach: Enabled property

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

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

Enables or disables a button object so you can or cannot use it, respectively. An object appears dimmed (greyed) when it is disabled.

Data type

Integer

Syntax

buttonobject.**Enabled** = *flag*

flag = *buttonobject*.**Enabled**

Legal values

Value	Description
TRUE	(Default) The object is enabled.
FALSE	The object is disabled; it appears dimmed.

Approach: EncloseLabel property

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

Sets or returns whether a border encloses a label.

Data type

Integer

Syntax

borderobject.**EncloseLabel** = *flag*

flag = *borderobject*.**EncloseLabel**

Legal values

Value	Description
FALSE	(Default) Do not enclose the label with a border.
TRUE	Enclose the label with a border.

Approach: Expand property

{button ,AL(`H_BODYPANEL_CLASS;H_FIELDBOX_CLASS;H_SUMMARYPANEL_CLASS;',0)} See list of classes

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

Sets or returns whether an object, such as a summary panel or fieldbox, expands when printing.

Data type

Integer

Syntax

objectname.**Expand** = *flag*

flag = *objectname*.**Expand**

Legal values

Value	Description
TRUE	(Default) Expand the object when printing.
FALSE	Do not expand the object when printing.

Approach: Font property

{button ,AL(`H_BUTTON_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Sets or returns the font object used for the object.

Data type

Font

Syntax

set objectname.**Font** = *fontobject*

set fontobject= objectname.**Font**

Legal values

Any font object.

Approach: FontName property

{button ,AL(`H_FONT_CLASS',1)} See list of classes

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

Sets or returns the name of the font. For example, you can change the font from Arial to Helvetica.

Data type

String

Syntax

fontobject.**FontName** = *stringexp*

stringexp = *fontobject*.**FontName**

Legal values

Arial is the default value. Any fonts currently installed on the computer are also legal values.

Approach: Green property

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

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

(Read-only) Returns the green component of an RGB value for an object.

Data type

Integer

Syntax

value = *colorobjectname*.**Green**

Legal values

0 represents no green and 255 represents the highest amount of green available.

Usage

Use the SetRGB method to change the settings of the Red, Green, and Blue properties at once.

Approach: Height property

{button ,AL(`H_BODYPANEL_CLASS;H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_HEADERFOOTERPANEL_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PANEL_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the height of an object in TWIPS.

Data type

Long

Syntax

objectname.**Height** = value

value = *objectname*.**Height**

Legal values

You can set the height of an object to the maximum height allowed by your printer.

Approach: HideMargins property

{button ,AL(`H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABEL_CLASS;',0)} See list of classes

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

Sets or returns whether the margins are displayed.

Data type

Integer

Syntax

viewname.**HideMargins** = *flag*

flag = *viewname*.**HideMargins**

Legal values

Value	Description
FALSE	(Default) Do not hide the margins.
TRUE	Hide the margins.

Approach: IsBeginOfData property

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

(read-only) Determines whether the record pointer is at the beginning of the DataSet.

Data type

Integer

Syntax

flag = *resultsetobject*.**IsBeginOfData**

Legal values

Value	Description
FALSE	The record pointer is not at the beginning of the DataSet.
TRUE	The record pointer is at the beginning of the DataSet.

Approach: IsChecked property

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

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

(read-only) Determines whether a checkbox is checked. Use the SetValue method to turn the checkbox on or off.

Data type

Integer

Syntax

flag = *checkboxobject*.**IsChecked**

Legal values

Value	Description
FALSE	(Default) The checkbox is not checked.
TRUE	The checkbox is checked.

Approach: IsConnected property

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

(read-only) Returns whether you have successfully connected to a server.

Data type

Integer

Syntax

flag = *connectobject*.**IsConnected**

Legal values

Value	Description
TRUE	There is a connection.
FALSE	There is not a connection.

Approach: IsEndOfData property

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

(read-only) Returns whether the record pointer is at the end of the DataSet.

Data type

Integer

Syntax

flag = *resultsetobject*.**IsEndOfData**

Legal values

Value	Description
TRUE	The record pointer is at the end of the DataSet.
FALSE	The record pointer is not at the end of the DataSet.

Approach: IsReadOnly property

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

(read-only) Returns whether a result set or database is read-only or read-write. A read-only result set cannot have records added, deleted, or changed.

Data type

Integer

Syntax

flag = *resultsetobject*.**IsReadOnly**

Legal values

Value	Description
TRUE	The result set or database is read-only.
FALSE	The result set or database is read-write.

Approach: IsResultSetAvailable property

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

(read-only) Determines whether the result set is available. An unavailable result set does not return any records.

Data type

Integer

Syntax

flag = *resultsetobject*.**IsResultSetAvailable**

Legal values

Value	Description
TRUE	The result set is available.
FALSE	The result set is not available.

Approach: Italic property

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

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

Sets or returns whether a font object is italicized.

Data type

Integer

Syntax

fontobject.**Italic** = *flag*

flag = *fontobject*.**Italic**

Legal values

Value	Description
FALSE	The font is not italic.
TRUE	The font is italic.

Approach: LabelAlignment property

{button ,AL(`H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the alignment between the label and a data-entry object, such as a checkbox or radio button.

Data type

Long

Syntax

objectname.**LabelAlignment** = *value*

value = *objectname*.**LabelAlignment**

Legal values

Value	Description
$tsAlignmentLeft	(Default) Aligns the label to the left edge of the object.
$ltsAlignmentHorizCenter	Aligns the label to the center of the object.
$ltsAlignmentRight	Aligns the label to the right edge of the object.

Approach: LabelFont property

{button ,AL(`H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the font for an object label. For example, you can change the label from Arial to Helvetica.

Data type

Font

Syntax

set objectname.**LabelFont** = *value*

set value = *objectname*.**LabelFont**

Legal values

You can set the property to any font object.

Approach: LabelPosition property

{button ,AL(`H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the position of the field label relative in relation to its associated field. For example, you can move the label from above to the left of the object.

Data type

Long

Syntax

objectname.**LabelPosition** = *value*

value = *objectname*.**LabelPosition**

Legal values

Value	Description
$tsPositionNone	The object does not have a label.
$ltsPositionTop	The label appears above the object.
$ltsPositionBottom	The label appears below the object.
$ltsPositionLeft	The label appears to the left of the object.
$ltsPositionRight	The label appears to the right of the object.

Approach: LabelText property

{button ,AL(`H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the text for a field label. For example, a field box name and its label are the same unless you change the text of the label.

Data type

String

Syntax

objectname.**LabelText** = *stringexp*

stringexp = *objectname*.**LabelText**

Legal values

You can set the property to any string up to ? characters.

Approach: Left property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the placement of the left edge of an object to the left edge of its container, measured in TWIPS. For example, the left edge of a fieldbox can be placed 20 TWIPS from the left edge of a form.

Data type

Long

Syntax

objectname.**Left** = *value*

value = *objectname*.**Left**

Legal values

If you set the object to a negative value, it hides the object.

Approach: Left property (Border class)

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

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

Sets or returns whether the left border of an object is displayed.

Data type

Integer

Syntax

borderobject.**Left** = *flag*

flag = *borderobject*.**Left**

Legal values

Value	Description
TRUE	(Default) Show the left border.
FALSE	Do not show the left border.

Approach: LineSpacing property

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

Sets or returns the amount of space between lines of text in a field box. For example, if single line spacing is too hard to read, you can increase it by another half or full line.

Data type

Long

Syntax

textobjectname.**LineSpacing** = *value*

value = *textobjectname*.**LineSpacing**

Legal values

Value	Description
$ItsLineSpacingSingle	(Default) Use single line spacing.
$ItsLineSpacingSingleAndHalf	Use a line and a half spacing.
$ItsLineSpacingDouble	Use two line spacing.

Approach: LineStyle property

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

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

Sets and returns the LineStyle object which specifies the style of a line; for example, the line color and pattern.

Data type

LineStyle

Syntax

set lineobject.**LineStyle** = *linestyleobject*

set linestyleobject = *lineobject*.**LineStyle**

Legal Values

You can set the property to any LineStyle object.

Approach: MacroClick property

{button ,AL(`H_BUTTON_CLASS;H_ELLIPSE_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the name of the macro to run when you click an object.

Data type

String

Syntax

objectname.**MacroClick** = *stringexp*

stringexp = *objectname*.**MacroClick**

Legal values

Any macro in the .APR file.

Approach: MacroDataChange property

{button ,AL(`H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_PICTUREPLUS_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the macro to run when the data in a data-entry object changes. For example, you can run a macro when a checkbox becomes checked.

Data type

String

Syntax

objectname.**MacroDataChange** = *macroname*

macroname = *objectname*.**MacroDataChange**

Legal values

Any macro in the .APR file.

Approach: MacroTabIn property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the name of the macro to run when you tab into an object.

Data type

String

Syntax

displayobject.**MacroTabIn** = *stringexp*

stringexp = *displayobject*.**MacroTabIn**

Legal values

Any macro in the .APR file.

Approach: MacroTabOut property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the name of the macro to run when you tab out of an object.

Data type

String

Syntax

displayobject.**MacroTabOut** = *stringexp*

stringexp = *displayobject*.**MacroTabOut**

Legal values

Any macro in the .APR file.

Approach: Magenta property

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

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

(read-only) Returns the magenta component of a CMYK value for an object.

Data type

Integer

Syntax

value = *colorobjectname*.**Magenta**

Legal values

Any integer from 0 to 255.

Usage

0 represents no magenta and 255 represents the highest amount of magenta available.

Approach: Name property

{button ,AL(`H_APPLICATION_CLASS;H_BUTTON_CLASS;H_CHARTVIEW_CLASS;H_CHECKBOX_CLASS;H_CROSSTAB_CLASS;H_DISPLAY_CLASS;H_DOCUMENT_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_ENVELOPE_CLASS;H_FIELDBOX_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_MAILINGLABEL_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_REPORT_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

Sets or returns the name of the current or specified object. You must specify unique names for each level of hierarchy. For example, all display objects in a panel must have unique names.

Data type

String

Syntax

stringexp = *objectname*.**Name**

objectname.**Name =** *stringexp*

Legal values

Any string that conforms to the object naming convention.

Approach: NamedStyle property

{button ,AL(`H_BODYPANEL_CLASS;H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_HEADERFOOTERPANEL_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_PANEL_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the name of the named style for an object.

Data type

String

Syntax

objectname.**NamedStyle** = *stringexp*

stringexp = *objectname*.**NamedStyle**

Legal values

You can set the property to any named style in the document.

Approach: NonPrinting property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns whether an object is visible when printing a form or report. For example, you can hide macro and script buttons when you print a form or report.

Data type

Integer

Syntax

objectname.**NonPrinting** = *flag*

flag = *objectname*.**NonPrinting**

Legal values

Value	Description
FALSE	(Default) Print the object.
TRUE	Do not print the object.

Approach: Orientation property

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

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

Sets or returns the orientation of the line graphic.

Data type

Long

Syntax

lineobject.**Orientation** = *value*

value = *lineobject*.**Orientation**

Legal values

Value	Description
$ItsOrientationPosSlope	(Default) Do not flip the line horizontally.
$ItsOrientationNegSlope	Flip the line horizontally.

Approach: Page property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

(read-only) Returns the page number of the form or report the current or specified object is on.

Data type

Integer

Syntax

value = *objectname*. **Page**

Legal values

Any integer between 1 and 5. The default value for the Page property is the page of the form or report the object is on.

Approach: Parent property (Application class)

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

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

(read-only) Returns the parent object of the current or specified application.

Data type

Application

Syntax

application = *applicationname*.**Parent**

Legal values

The default value for the Parent property is Approach itself.

Approach: Parent property (Document class)

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_DOCWINDOW_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABEL_CLASS;H_REPORT_CLASS;H_TABLE_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

(read-only) Returns the parent object of the current or specified document.

Data type

Document

Syntax

application = *documentname*.**Parent**

Legal values

The default value for the Parent property of the document class is the application.

Approach: Parent property

{button ,AL(`H_BODYPANEL_CLASS;H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_HEADERFOOTERPANEL_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PANEL_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECTCLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

(read-only) Returns the parent object of the current object.

Data type

Variant

Syntax

parentobject = *objectname*.**Parent**

Legal values

The default value for the Parent property is the current object.

Approach: Pattern property

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

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

Sets or returns the pattern of backgrounds and lines. For example, you can change the pattern of a line from solid to dashed.

Data type

Long

Syntax

objectname.**Pattern** = *value*

value = *objectname*.**Pattern**

Legal values

Value	Description
$ItsBorderPatternSolid	(Default) Display a solid line.
$ItsBorderPatternDouble	Display a double solid line.
$ItsBorderPatternDot	Display a dotted line.
$ItsBorderPatternDashed	Display a dashed line.
$ItsBorderPatternDashDot	Display an alternating dashed and dotted line.
$ItsBorderPatternDashDotDot	Display an alternating dashed and two dots line.

Approach: Position property

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

Sets or returns the position of a picture in a PicturePlus field relative to its frame. The frame is divided into a 3 x 3 grid, providing nine possible positions for the picture.

Data type

Long

Syntax

ppobjectname.**Position** = *value*

value = *ppobjectname*.**Position**

Legal values

Value	Description
$ItsPositionTopLeft	(Default) The picture is in the top left position (1 in the 9 block grid).
$ItsPositionTop	The picture is in the top center position (2 in the 9 block grid).
$ItsPositionTopRight	The picture is in the top right position (3 in the 9 block grid).
$ItsPositionLeft	The picture is in the middle left position (4 in the 9 block grid).
$ItsPositionCenter	The picture is in the center position (5 in the 9 block grid).
$ItsPositionRight	The picture is in the middle right position (6 in the 9 block grid).
$ItsPositionBottomLeft	The picture is in the bottom left position (7 in the 9 block grid).
$ItsPositionBottom	The picture is in the bottom center position (8 in the 9 block grid).
$ItsPositionBottomRight	The picture is in the bottom right position (9 in the 9 block grid).

Approach: Stretch property

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

Sets or returns whether the picture in a PicturePlus field stretches if it is too small for the frame.

Data type

Integer

Syntax

ppobjectname.**Stretch** = *flag*

flag = *ppobjectname*.**Stretch**

Legal values

Value	Description
FALSE	(Default) The picture is not stretched to fit the PicturePlus field.
TRUE	The picture is stretched to fit the PicturePlus field

Approach: Query property

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

Sets or returns the query you want to use when you retrieve the data.

Data type

Query

Syntax

set resultsetname.**Query** = *queryname*

set queryname = *resultsetame*.**Query**

Legal Values

You can set this property to any query object.

Approach: ReadOnly property

{button ,AL(`H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_PICTUREPLUS_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns if a data-entry object is read-only or read/write. For example, you cannot change a read-only checkbox selection.

Data type

Integer

Syntax

objectname.**ReadOnly** = *flag*

flag = *objectname*.**ReadOnly**

Legal values

Value	Description
FALSE	(Default) The object is read-write.
TRUE	The object is read-only.

Approach: Red property

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

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

(read-only) Returns the red component of an RGB value for an object.

Data type

Integer

Syntax

value = *colorobjectname*.**Red**

Legal values

Any integer from 0 to 255.

Usage

0 represents no red and 255 represents the highest amount of red available.

Approach: Reduce property

{button ,AL(`H_BODYPANEL_CLASS;H_FIELDBOX_CLASS;H_SUMMARYPANEL_CLASS;',0)} See list of classes

Sets or returns whether an object, such as a bodypanel or fieldbox, is reduced when printing.

Data type

Integer

Syntax

objectname.**Reduce** = *flag*

flag = *objectname*.**Reduce**

Legal values

Value	Description
FALSE	(Default) Do not reduce the object when printing.
TRUE	Reduce the object when printing.

Approach: Relief property

{button ,AL(`H_CHECKBOX_CLASS;H_FONT_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

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

Sets or returns the appearance of a font. For example, you can make the font look as if it is sticking out of the page by using a 3D raised relief.

Data type

Long

Syntax

fontobject.**Relief** = *value*

value = *fontobject*.**Relief**

Legal values

Value	Description
$ItsReliefNone	(Default) The font appears normally.
$ItsReliefRaised	The font has a 3D raised effect.
$ItsReliefLowered	The font has an indented effect.

Approach: Right property (Border class)

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

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

Sets or returns whether the right side border of an object is displayed.

Data type

Integer

Syntax

borderobject.**Right** = *flag*

flag = *borderobject*.**Right**

Legal values

Value	Description
TRUE	(Default) Show the right border.
FALSE	Do not show the right border.

Approach: ShadowColor property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the color to use for the shadow of an object.

Data type

Color

Syntax

set objectname.**ShadowColor** = *color*

set color = *objectname*.**ShadowColor**

Legal values

The default color for the ShadowColor property is transparent. You can set this property to any color object.

Approach: ShowArrow property

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

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

Sets or returns whether the drop-down arrow is displayed in a drop-down box. For example, you may want to turn the drop-down arrow off when you have only one list item in a drop-down box.

Data type

Integer

Syntax

ddboxname.**ShowArrow** = *flag*

flag = *ddboxname*.**ShowArrow**

Legal values

Value	Description
TRUE	(Default) Show the drop-down arrow.
FALSE	Do not show the drop-down arrow.

Approach: ShowAsDialog property

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

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

Sets or returns whether the form is shown as a dialog box.

Data type

Integer

Syntax

formname.**ShowAsDialog** = *flag*

flag = *formname*.**ShowAsDialog**

Legal values

Value	Description
FALSE	(Default) Do not show the form as a dialog box.
TRUE	Show the form as a dialog box.

Approach: ShowInPreview property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns whether an object is displayed when previewing a view.

Data type

Integer

Syntax

displayname.**ShowInPreview** = *flag*

flag = *displayname*.**ShowInPreview**

Legal values

Value	Description
TRUE	(Default) Show the object while in PrintPreview.
FALSE	Do not show the object while in PrintPreview.

Approach: Size property

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

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

Sets or returns the size of the font type in points. For example, you can change the Font size from 6 points to 18 points.

Data type

Single

Syntax

fontobject.**Size** = *value*

value = *fontobject*.**Size**

Legal values

10 points is the default value for the Size property. Other legal values depend on the font.

Approach: SlideLeft property

{button ,AL(`H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns whether an object is arranged so that it closes gaps with objects to the left of it or not when printing. For example, in mailing labels you may have a field for cities which is 30 characters long. If you enter a city which is only 10 characters long, there will be twenty blank spaces between the city and the state when you print the view. If you specify to slide the state field object to the left, the gap of 20 spaces is closed up.

Data type

Integer

Syntax

displayobject.**SlideLeft** = *flag*

flag = *displayobject*.**SlideLeft**

Legal values

Value	Description
FALSE	(Default) Do not slide the object to the left when printing.
TRUE	Slide the object to the left when printing.

Approach: SlideUp property

{button ,AL(`H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets and returns whether an object is arranged so that it closes gaps with objects above it or not. For example, you may have a field box object which is 300 characters long, and then a picture object underneath it. If you only enter 100 characters into the textbox, there will be 200 blank spaces between the text and the picture when you print the view. if you specify to slide the picture object up, the gap of 200 blank spaces is closed up.

Data type

Integer

Syntax

displayobject.**SlideLeft** = *flag*

flag = *displayobject*.**SlideLeft**

Legal values

Value	Description
FALSE	(Default) Do not slide the object up when printing.
TRUE	Do slide the object up when printing.

Approach: SQL property

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

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

Sets or returns whether an SQL statement is set or retrieved.

SQL statements may be very long, so care must be taken when reading the SQL statement. When an SQL statement is stored, it is automatically checked to insure that it is syntactically correct. There is no practical limit to the SQL statement that can be written. The SQL statement is not executed until an Execute is performed. The SQL statement that may be executed depends on a user's access rights. A user may not have authorization to access some tables and/or columns. This property is mutually exclusive with the TableName property. If an SQL statement is specified, ADO will execute the SQL statement.

Data type

String

Syntax

queryobject.**SQL** = *stringexp*

stringexp = *queryobject*.**SQL**

Legal values

There is no default value for the SQL property. You can set this property to any SQL SELECT string.

Approach: Strikethrough property

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

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

Sets or returns the text attirbute for a font to Strikethrough.

Data type

Integer

Syntax

fontobject.**Strikethru** = *flag*

flag = *fontobject*.**Strikethru**

Legal values

Value	Description
FALSE	(Default) Do not cross out the font.
TRUE	Cross out the font.

Approach: TabOrder property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the order in which objects get the focus when using the TAB key to move around a form. The default tab order is based on the order in which objects are created.

Data type

Integer

Syntax

displayobject.**TabOrder** = *value*

value = *displayobject*.**TabOrder**

Legal values

You can set this property to any integer between 1 and the number of objects in the view.

Approach: TabStop property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns whether an object can get the focus by pressing the TAB key. For example, if you have a form with a picture field, and you do not want users to tab into the picture, turn off the TabStop property for that picture field.

Data type

Integer

Syntax

displayobject.**TabStop** = *flag*

flag = *displayobject*.**TabStop**

Legal values

Value	Description
TRUE	(Default, except for circle, line, rectangle, roundrect, and field box objects.) Allow for tabbing into the object.
FALSE	Do not allow tabbing into the object.

Approach: Text property

{button ,AL(`H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_TEXTBOX_CLASS;',0)}
See list of classes

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

Sets or returns the text value of the data in drop-down boxes, field boxes, listboxes and others.

Data type

String

Syntax

objectname.**Text** = *stringexp*

stringexp = *objectname*.**Text**

Legal values

You can set this property to any string up to ? characters.

Approach: Text property (Button class)

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

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

Sets or returns the text to be displayed on a button.

Data type

String

Syntax

buttonobject.**Text** = *stringexp*

stringexp = *buttonobject*.**Text**

Legal values

There is no default value for the Text property. You can set this property to any string up to ? characters.

Approach: Top property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sets or returns the placement of the top edge of an object to the top edge of its container, measured in TWIPS. For example, the top edge of a fieldbox can be placed 30 TWIPS from the top edge of the form.

Data type

Long

Syntax

objectname.**Top** = *value*

value = *objectname*.**Top**

Approach: Top property (Border class)

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

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

Sets or returns whether the top border of an object is displayed or not.

Data type

Integer

Syntax

borderobject.**Top** = *flag*

flag = *borderobject*.**Top**

Legal values

Value	Description
TRUE	(Default) Show the top border.
FALSE	Do not show the top border.

Approach: Type property

{button ,AL(`H_BODYPANEL_CLASS;H_BUTTON_CLASS;H_CHARTVIEW_CLASS;H_CHECKBOX_CLASS;H_CROSSTAB_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_ENVELOPE_CLASS;H_FIELDBOX_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_HEADERFOOTERPANEL_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_MAILINGLABEL_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_REPORT_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

(read-only) Returns the type of display object this object is.

Data type

Long

Syntax

value = *objectname*.**Type**

Legal values

For the legal values you can use for this property, see the list of enumerators.

Approach: Underline property

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

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

Sets or returns the text attribute of a font to underline.

Data type

Integer

Syntax

fontobject.**Underline** = *flag*

flag = *fontobject*.**Underline**

Legal values

Value	Description
FALSE	(Default) Do not underline the font.
TRUE	Underline the font.

Approach: UncheckedValue property

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

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

Sets or returns the value you want to store in a field if the checkbox is unchecked.

Data type

String

Syntax

*checkboxobject***.UncheckedValue** = *stringexp*

stringexp = *checkboxobject***.UncheckedValue**

Legal values

There is no default value for the UncheckedValue property.You can set this property to any string up to ? characters.

Approach: UserID property

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

Identifies who is currently connected to a database.

Data type

String

Syntax

connectobject.**UserID** = *stringexp*

stringexp = *connectobject*.**UserID**

Legal values

There is no default value for the UserID property. You can set this property to any string up to ? characters.

Approach: Value property (CheckBox class)

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

Sets or returns the value of the checkbox based on whether it is checked or unchecked.

Data type

String

Syntax

checkboxobject.**Value** = *stringexp*

stringexp = *checkboxobject*.**Value**

Legal values

There is no default value for the Value property (CheckBox). You can set this property to any value that is legal for the field, determined by its properties.

Approach: Value property (RadioButton class)

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

Sets or returns the value of the currently selected radio button.

Data type

String

Syntax

radiobuttonobject.**Value** = *stringexp*

stringexp = *radiobuttonobject*.**Value**

Legal values

There is no default value for the Value (RadioButton) property. You can set this property to any value that is legal for the field, determined by its properties.

Approach: Visible property

{button ,AL(`H_APPLICATION_CLASS;H_BUTTON_CLASS;H_CHARTVIEW_CLASS;H_CHECKBOX_CLASS;H_CROSSTAB_CLASS;H_DISPLAY_CLASS;H_DOCWINDOW_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_ENVELOPE_CLASS;H_FIELDBOX_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_MAILINGLABEL_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPORT_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See related topics

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

Sets or returns whether an object is visible or not.

Data type

Integer

Syntax

objectname.**Visible** = *flag*

flag = *objectname*.**Visible**

Legal values

Value	Description
TRUE	(Default) The object is visible.
FALSE	The object is not visible.

Approach: Width property

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LINESTYLE_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See related topics

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

Sets or returns the width, in TWIPS, of an object.

Data type

Long

Syntax

objectname.**Width** = *value*

value = *objectname*.**Width**

Approach: Width property (Border class)

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

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

Sets or returns the width of a border line.

Data type

Long

Syntax

borderobject.**Width** = *value*

value = *borderobject*.**Width**

Legal values

Value	Description
$ItsBorderWidthNone	There is no border.
$ItsBorderWidthThin	The border is a thin line.
$ItsBorderThick	The border is a thick line.

Approach: Yellow property

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

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

(Read-only) Returns the yellow component of a CMYK value for an object.

Data type

Integer

Syntax

value = *colorobjectname*.**Yellow**

Legal values

Any integer from 0 to 255.

Usage

0 represents no yellow and 255 represents the highest amount of yellow available.

Approach: Activate method

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

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

Makes the specified document active.

Syntax

documentobjectname.**Activate**

Parameters

None

Return values

None

Approach: Add method (Collection class)

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

Adds an object to this collection.

Syntax

index = *collectionobjectname*.**AddCollection**(*object*)

Parameters

object

An object added to the collection.

Return values

Value	Description
Index	Returns an integer representing the index for this collection.

Approach: AddColumn method

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

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

Adds a column to a worksheet.

Syntax

integer = *worksheetobjectname*.**AddColumn**(*fieldname* | *[columnname, insertposition*])

Parameters

fieldname

A string representing the name of a field.

columnname

(Optional) A string representing the name of a column.

insertposition

(Optional) A string representing the position to insert the column in the worksheet.

Return values

Value	Description
TRUE	A new column was added to the worksheet.
FALSE	A new column was not added to the worksheet.

Approach: AddRow method

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

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

Adds a new row to a database. AddRow can be used to add a row to a table or the result set. The row contains no valid data for any column. The row must be populated with data using SetValue. The row is not updated until an UpdateRow is performed.

Syntax

resultsetobjectname.**AddRow**

integer = *resultsetobjectname*.**AddRow**()

Parameters

None

Return values

Value	Description
TRUE	A new row was added to the table or result set.
FALSE	A new row was not added to the table or result set.

Approach: BringToFront method

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_ELLIPSE_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Brings a display object to the front of a panel. For example, you can have a circle around a group of radio buttons. However, to click the buttons, they must be in front of the circle object.

Syntax

objectname.**BringToFront**

Parameters

None

Return values

None

Approach: Cascade method

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

Cascades the windows. Using this LotusScript command is the same as choosing Window - Cascade from the menu.

Syntax

Set *applicationwindowobjectname*.**Cascade**

Parameters

None

Return values

None

Approach: Close method

{button ,AL(`H_APPLICATIONWINDOW_CLASS;H_DOCWINDOW_CLASS;H_WINDOW_CLASS;',0)} See list of classes

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

Closes the window. Using this LotusScript command is the same as if you closed using the system icon.

Syntax

Set *applicationwindowobjectame*. **Close**

Parameters

None

Return values

None

Note

If a form as Dialog is operating, closing the Doc Window will cause the Form As Dialog to close.

Approach: Close method (ResultSet class)

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

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

Closes a result set and commits or rolls back changes. The entire result set is cleared after a close.

Syntax

resultsetobjectname.**Close**

integer = *resultsetobjectname*.**Close**

Parameters

None

Return values

Value	Description
TRUE	The result set was closed.
FALSE	The result set was not closed.

Approach: CloseWindow method

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

Close the application's window. Optionally, (though not by default) you can exit the application as though you had used the APP.Quit() command.

Syntax

applicationobjectname.**CloseWindow**([*quit*])

Parameters

quit

(Optional) A variant representing the action to be taken.

Return values

None

Approach: ConnectTo method

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

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

Connects to a data source.

Syntax

integer = *connectionobjectname*.**ConnectTo**(*datasourcename,* [*userid, password*] [*server*])

Parameters

datasourcename

A string representing the name of the data source to which you are connecting. See the list of data sources to which Approach can connect.

userid

(Optional) A string representing a user id.

server

(Optional) A string representing the server for the data source or the path location for the data source.

Return values

Value	Description
TRUE	You connected to the data source.
FALSE	You did not connect to the data source.

Usage

The *userid* and *password* are saved for this data source during this script session so that subsequent connection attempts to the same data source will be automatically performed.

Note

The following methods (ListTables, ListFields, and Execute) let you change the default path specified in ConnectTo. For example:

ListTables(*newpath*)

Approach: CreateResultSet method

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

The data that currently displays onscreen will be used to create a ResultSet object.

Syntax

Set *ResultSetObject* = *tableobjectname*.**CreateResultSet**()

Parameters

None

Return values

Value	Description
ResultSet	Returns a ResultSet object created from the data that is currently onscreen.

Approach: DeletePage method

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

Removes a page from a form.

Syntax

integer = *formobjectname*.**DeletePage**(*pagenumber*)

Parameters

pagenumber

The page number to be deleted.

Return values

Value	Description
TRUE	The page was successfully deleted.
FALSE	The page was not successfully deleted.

Approach: DeleteRow method

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

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

Deletes a record represented by the current row in a result set from a database. DeleteRow, like UpdateRow, may not be possible if the found set or database is read-only.

Syntax

resultsetobjectname.**DeleteRow**

integer = *resultsetobjectname*.**DeleteRow**()

Parameters

None

Return values

Value	Description
TRUE	The record in the database was deleted.
FALSE	The record in the database was not deleted.

Approach: Disconnect method

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

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

Disconnects you from a data source.

Syntax

connectionobjectname.**Disconnect**

integer = *connectionobjectname*.**Disconnect**()

Parameters

None

Return values

Value	Description
TRUE	You disconnected from the data source.
FALSE	You did not disconnect from the data source.

Approach: DoMenuCommand method

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

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

Executes a command on a menu. The menus are defined as constants.

Syntax

integer = *applicationwindowobjectname*.**DoMenuCommand**(*command*)

Parameters

command

A short representing a menu command.

Return values

Value	Description
TRUE	The menu command was executed.
FALSE	The menu command was not executed.

Approach: DoVerb method

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

Performs an OLE operation (verb) on an OLE object. The OLE server container determines the verbs available for the object.

Syntax

integer = *OLEobjectname*.**DoVerb**(*verbnumber*)

Parameters

verbnumber

A number representiing the verb to perform. Zero (0) is the primary verb, corresponding to the action performed when double-clicking the object. Click the object with the right mouse button to open a list of the available verbs. The primary verb (0) is listed first. Use verbnumber = 1 to perform the second verb in the list, and so on.

If verbnumber does not correspond to a verb available for the object, the primary verb is performed.

Return values

Value	Description
TRUE	The verb was performed.
FALSE	The verb was not performed.

Approach: Execute method

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

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

Executes the query specified in the query property of the result set against the connection specified in the connection property of the query.

Syntax

resultsetobjectname.**Execute**

integer = *resultsetobjectname*.**Execute**()

Parameters

None

Return values

Value	Description
TRUE	The SQL statement was executed.
FALSE	The SQL statement was not executed.

Usage

This method is used to execute a query, refresh a result set in a query, or execute any other valid SQL statement.

Note

Execute lets you change the default path specified in the ConnectTo method. For example:

Execute(*newpath*)

Approach: FieldExpectedDataType method

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

Sets or returns the expected data type. The expected data type creates a conversion between the actual data type for a column value in a result set and the data type expected by the application. When a field value is read (fetched) from the database, no specific data conversion routines are assigned. All field values are returned as they appear in the database. A real value will return as real, integers as integers, and strings as strings.

Syntax

datatype = *resultsetobjectname*.**FieldExpectedDataType**(*columnname* | [*columnid*]*, datatype*)

Parameters

columnname

A string representing the name of a column.

columnid

(Optional) An integer representing the ID value of the column. Use as an alternate way to specify the column, if the name is unknown.

datatype

An integer representing the type of data the field expects to be returned. For a list of data types, see "Return Values" below.

Return values

Value	Description
DB_BOOL	The data type is an integer.
DB_BOOLEAN	The data type is a boolean.
DB_CHAR	The data type is a string.
DB_DATE	The data type is a date.
DB_DOUBLE	The data type is a double.
DB_LONG	The data type is a long.
DB_SHORT	The data type is a short.
DB_TIME	The data type is a time.
DB_UNDEFINED	Resets the data type to default.

Usage

To determine the database data type, use FieldNativeDataType, FieldSize, and FieldInfo. FieldNativeDataType returns the data type as stored in the database for this column. FieldSize returns the field size of the database. FieldInfo returns the field information as a Field object and contains all of the known field information.

Approach: FieldID method

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

Returns the column id value for a field. The field name is searched in the result set and corresponds to the relative column number, that is, the first column has a field id of 1, the second column has a field id of 2, and so on.

Syntax

idvalue = *resultsetobjectname*.**FieldID**(*columnname*)

Parameters

columnname

A string representing the name of a column.

Return values

Value	Description
ColumnId	An integer representing the field value of the specified column.

Usage

FieldID is not necessary when walking through all of the fields in a result set. The field id is required only when a field's name is known, but its column id is not. It provides a good level of defensive programming. For example, suppose that in a result set the social security number field is retrieved, usually as column 2. If the result set were ever changed, the field's relative position, and consequently its ID, would change. A defensive programming practice would always determine dynamically the field ID, as follows:

```
ssn_id% = object. FieldID("SSN")
text = GetValue(ssn_id%)
```

Or

```
text = GetValue( object.FieldID( "SSN" ))
```

Approach: FieldName method

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

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

Returns the name of a specified field.

Syntax

fieldname = *resultsetobjectname*.**FieldName**(*columnid*)

Parameters

columnid

An integer representing the id value of a field.

Return values

Value	Description
FieldName	A string representing the name of the specified field.

Approach: FieldNativeDataType method

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

Returns the native data type for the specified field. The native data type is the data type of the field as stored in the database. The data type of the data value returned to script is the value set by the expected data type. This value may be retrieved by using FieldInfo.

Syntax

fieldvalue = *resultsetobjectname*.**FieldNativeDataType**(*columnname* | [*columnid*])

Parameters

columnname

A string representing the name of a column.

columnid

(Optional) An integer representing the id value of the column. Use as an alternate way to specify the column, if the name is unknown.

Return values

Value	Description
SQL_BINARY	The data type is a binary.
SQL_BIT	The data type is an integer.
SQL_CHAR	The data type is a string.
SQL_DOUBLE	The data type is a double.
SQL_TIME	The data type is a time.
SQL_VARBINARY	The data type is a pictureplus.
SQL_VARCHAR	The data type is a memo.

Approach: FieldSize method

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

Returns the maximum size of data for a field.

Syntax

fieldsize = *resultsetobjectname*.**FieldSize**(*columnname* | [*columnid*])

Parameters

columnname

A string representing the name of a column.

columnid

(Optional) An integer representing the id value of the column. Use as an alternate way to specify the column, if the name is unknown.

Return values

Value	Description
fieldsize	A long representing the maximum size of the field.

Approach: FillField method

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

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

Fills a field with the value specified in the current found set. Using this LotusScript command is the same as executing the Fill Field command from the context menu.

Syntax

integer = *docwindowobjectname*.**FillField**(*fieldname, value*)

Parameters

fieldname

A string representing the specified field.

value

A variant representing the value you want to use to update the field.

Return values

Value	Description
TRUE	The specified field was filled with the specified value.
FALSE	The specified field was not filled with the specified value.

Approach: FirstRecord method

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

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

Makes the first record in the found set the current record. Using this LotusScript command is the same as clicking the First Record SmartIcon.

Syntax

docwindowobjectname.**FirstRecord**

Parameters

None

Return values

None

Approach: FirstRow method

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

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

Sets the current row to be the first row in the result set. It is equivalent to setting CurrentRow to 1.

Syntax

resultsetobjectname.**FirstRow**

integer = *resultsetobjectname*.**FirstRow**()

Parameters

None

Return values

Value	Description
TRUE	The current row was set to be the first row.
FALSE	The current row was not set to be the first row.

Approach: GetColorFromRGB

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

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

Returns a color object when given an RGB value. Many of the RGB values are defined as constants.

Syntax

color = *applicationobjectname*.**GetColorFromRGB**(*rgbvalue*)

Parameters

rgbvalue

A long representing an RGB value.

Return values

Value	Description
Color	Returns a color object.

Approach: GetError method

{button ,AL(`H_CONNECTION_CLASS;H_QUERY_CLASS;H_RESULTSET_CLASS;',0)} See list of classes

Returns the error code value for an object.

Syntax

integer = *objectname*.**GetError**()

Parameters

None

Return values

Returns the error

Approach: GetErrorMessage method

{button ,AL(`H_CONNECTION_CLASS;H_QUERY_CLASS;H_RESULTSET_CLASS;',0)} See list of classes

Returns a short text message associated with an error code value.

Syntax

messagetext = *objectname*.**GetErrorMessage**(*errorvalue*)

Parameters

errorvalue

An integer representing the error code value for which you want to get the text message. If the *error value* is missing, the error message for the last error encountered is returned. The *error value* field can contain DB_LASTERROR to refer to the last error code value.

Return values

Value	Description
MessageText	A string representing the text for the specified error code value.

Approach: GetExtendedErrorMessage method

{button ,AL(`H_CONNECTION_CLASS;H_QUERY_CLASS;H_RESULTSET_CLASS;',0)} See list of classes

Returns a long text message associated with an error code value.

Syntax

messagetext = *objectname*.**GetExtendedErrorMessage**(*errorvalue*)

Parameters

errorvalue

An integer representing the error code value for which you want to get the text message. If the *error value* is missing, the error message for the last error encountered is returned. The *error value* field may contain DB_LASTERROR to refer to the last error code.

Return values

Value	Description
MessageText	A string representing the text for the specified error code value.

Approach: GetFieldFormula method

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

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

Returns the formula for the field specified in the FieldName parameter. The formula returned is a duplication of the formula in Field Definition.

Syntax

string = *tableobjectname*.**GetFieldFormula**(*fieldname*)

Parameters

fieldname

A string representing the formula in FieldName.

Return values

Value	Description
Formula	A string that returns the formula in the Field Definition.

Approach: GetFieldOptions method

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

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

Returns the options for the field specified in the FieldName parameter. The options returned are a text string listing of the options for the field. It is a duplication of the Formula/Options column in Field Definition.

Syntax

string = *tableobjectname*.**GetFieldOptions**(*fieldname*)

Parameters

fieldname

A string representing the field in the FieldName.

Return values

Value	Description
Options	Returns a string representing the options for the field specified in the FieldName parameter.

Approach: GetFieldSize method

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

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

Returns the size of the field specified in the FieldName parameter. Text and numeric fields are the only fields that return a value greater than 0. All other data types return a value of 0.

Syntax

integer = *tableobjectname*.**GetFieldSize**(*fieldname*)

Parameters

fieldname

A string representing the field specified in the FieldName parameter.

Return values

Value	Description
FieldSize	An integer that returns the size of the field specified in FieldName.

Approach: GetFieldType method

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

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

Returns the type of field specified in the FieldName parameter. The possible return values are: Boolean, Calculated, Date, Memo, Numeric, PicturePlus, Text, Time, and Variable.

Syntax

long = *tableobjectname*.**GetFieldType**(*fieldname*)

Parameters

fieldname

A string representing the type of field specified in FieldName.

Return values

Value	Description
Long	Returns a field's data type.

Approach: GetHandle method

{button ,AL(`H_APPLICATIONWINDOW_CLASS;H_DOCWINDOW_CLASS;H_WINDOW_CLASS;',0)} See list of classes

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

Returns a window handle to the window.

Syntax

long = *windowobjectname*.**GetHandle**

Parameters

None

Return values

Value	Description
Long	Returns a window handle.

Approach: GetParameter method

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

Returns the last value set for a specified parameter. A parameter name or its ordinal value can be used. If the specified parameter is invalid, you get an error message and no value is returned.

Syntax

parametervalue = *resultsetobjectname*.**GetParameter**(*parametername* | *ordinalposition*)

Parameters

parametername

A string representing the name of the parameter to get. You can also use the ordinal position to get a parameter.

ordinalposition

An integer representing the ordinal position of the parameter to get. Use as an alternate way to point to a parameter, if the name is unknown.

Return values

Value	Description
ParameterValue	A variant representing the value of the parameter.

Usage

Parameters are used in an SQL statement. SetParameter is used to set the replacement value of the parameter. Unlike ODBC and standard SQL, parameters can actually appear anywhere in the SQL statement.

Approach: GetParameterName method

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

Returns a parameter name contained in an SQL statement without its surrounding question marks.

Syntax

parametername = *resultsetobjectname*.**GetParameterName**(*parameterindex*)

Parameters

parameterindex

A long representing the index value of the parameter. The parameter names contained in an SQL statement can be retrieved using *parameterindex* as an index.

Return values

Value	Description
ParameterName	A string representing the name of the specified parameter.

Usage

The parameter names can be used to prompt a user.

Approach: GetRGB method

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

Returns the RGB value of the color.

Syntax

long = *colorobjectname*.**GetRGB**()

Parameters

None

Return values

Value	Description
Long	Returns the RED, GREEN, and BLUE values of the object's color

Approach: GetTableByName method

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

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

Syntax

table = *documentobjectname*.**GetTableByName**(*tablename*)

Parameters

tablename

A string representing the name of the selected table.

Return values

Value	Description
Table	Returns the selected table.

Approach: GetText method

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

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

Returns the text in the current row for the specified column. If there is no current row, then an empty string is returned.

Syntax

string = *worksheetobjectname*.**GetText**([*column*])

Parameters

column

(Optional) A string representing the name of a column.

Return values

Value	Description
Data	A string representing the text returned from the current row.

Approach: GetValue method

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

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

Returns the value of the column referenced by *column_id* in the current row and copies it to the *variable* field. The data type of the returned data is determined by the expected data type value set for this column by the FieldExpectedDataType property. For example, the expected value can be set to return a real value for an integer numeric field.

Syntax

value = *resultsetobjectname*.**GetValue**(*columnname* | [*columnid*])

Parameters

columnname

A string representing the name of a column.

columnid

(Optional) An integer representing the id value of the column. Use as an alternate way to specify the column, if the name is unknown.

Return values

Value	Description
value	The value of the specified column, or field.

Approach: InsertAfter method

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

InsertAfter places the object after the one specified and before all others in the hierarchy layer.

Syntax

integer = *objectname*.**InsertAfter**(*object*)

Parameters

object

An object from the Display class representing the object to be inserted.

Return values

Value	Description
TRUE	The object is inserted after the display object.
FALSE	The object is not inserted after the display object.

Approach: IsCommandChecked method

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

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

Returns TRUE or FALSE depending on if the menu command is checked. TRUE means the command is checked. The commands that can be checked are defined as constants.

Syntax

integer = *applicationwindowobjectname*.**IsCommandChecked**(*command*)

Parameters

command

A short representing the command to be evaluated by IsCommandChecked.

Return values

Value	Description
TRUE	The menu command was checked.
FALSE	The menu command was not checked.

Approach: IsCommandEnabled method

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

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

Returns TRUE or FALSE indicating whether or not the command is enabled. TRUE means the command is available. The commands that can be enabled are defined as constants.

Syntax

integer = *applicationwindowobjectname*.**IsCommandEnabled**(*command*)

Parameters

command

A short representing the command to be evaluated .

Return values

Value	Description
TRUE	The command is available.
FALSE	The command is not available.

Approach: IsEmpty method

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

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

Tests the value of a collection to determine if it is empty.

Syntax

integer = *basecollectionobjectname*.**IsEmpty**()

Parameters

None

Return values

Value	Description
TRUE	The collection was empty.
FALSE	The collection was not empty.

Approach: LastRecord method

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

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

Makes the last record in the found set the current record. Using this LotusScript command is the same as clicking the Last Record SmartIcon.

Syntax

docwindowobjectname.**LastRecord**

Parameters

None

Return values

None

Approach: LastRow method

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

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

Sets the current row to be the last row in the result set.

Syntax

resultsetobjectname.**LastRow**

integer = *resultsetobjectname*.**LastRow**()

Parameters

None

Return values

Value	Description
TRUE	The current row was set to be the last row.
FALSE	The current row was not set to be the last row.

Approach: ListDataSources method

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

Returns the list of all registered data sources as an array.

Syntax

stringarray = *connectionobjectname*.**ListDataSources**()

Parameters

None

Return values

Value	Description
StringArray	An array of strings listing the registered data sources.

Approach: ListFields method

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

Returns an array of field names. Each name corresponds to the name of a field in the referenced table.

Syntax

variantarray = *connectionobjectname*.**ListFields**(*tablename*)

Parameters

tablename

A string representing the name of a table. The *tablename* cannot be blank.

Return values

Value	Description
Fields	An array of variants representing the fields in a table.

Note

For dBASE files, you must use the complete path. For example:

```
ListFields("c:\testdb.dbf")
```

ListFields lets you change the default path specified in the ConnectTo method. For example:

```
ListTables(newpath)
```

Approach: ListTables method

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

Returns the names of the available database tables in a data source. If the data source name is missing, a list of database tables for the currently connected data source is returned.

Syntax

variantarray = *connectionobjectname*.**ListTables**(*datasourcename*)

Parameters

datasourcename

A string representing the name of the data source.

Return values

Value	Description
VariantArray	A variant representing an array of tables.

Note

ListTables does not apply to dBASE files.

ListTables lets you change the default path specified in the ConnectTo method. For example:

ListTables(*newpath*)

Approach: MakeNamedStyle method

{button ,AL(`H_BODYPANEL_CLASS;H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_HEADERFOOTERPANEL_CLASS;H_LINE_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PANEL_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_REPEATINGPANEL_CLASS;H_ROUNDRECT_CLASS;H_SUMMARYPANEL_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Creates a named style from the object's attributes. You specify a name for the named style.

Syntax

integer = *objectname*.**MakeNamedStyle**(*stylename*)

Parameters

stylename

A string representing the name of the named style being created.

Return values

Value	Description
TRUE	The named style was created.
FALSE	The named style was not created.

Approach: Maximize method

{button ,AL(`H_APPLICATIONWINDOW_CLASS;H_DOCWINDOW_CLASS;H_WINDOW_CLASS;',0)} See list of classes

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

Maximizes the window. Using this LotusScript command is the same as if you maximized using the system icon.

Syntax

windowname.**Maximize**

Parameters

None

Return values

None

Approach: Merge method

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

Each element in the Collection class is added to the current collection. The method returns the size (count) of the updated collection. No attempt is made to ensure uniqueness. If both collections contain the same object, it will appear in the merged collection twice.

Syntax

integer = *collectionobjectname*.**Merge**(*collection*)

Parameters

collection

A basecollection representing each element in the Collection.

Return values

Value	Description
TRUE	Each element in the Collection was merged with the current collection.
FALSE	Each element in the Collection was merged with the current collection.

Approach: Minimize method

{button ,AL(`H_APPLICATIONWINDOW_CLASS;H_DOCWINDOW_CLASS;H_WINDOW_CLASS;',0)} See list of classes

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

Minimizes the window. Using this LotusScript command is the same as minimizing using the system icon.

Syntax

windowname.**Minimize**

Parameters

None

Return values

None

Approach: New method (Button class)

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

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

Creates a new button.

Syntax

Set *buttonname* = **New Button** (*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the Button is placed. Use to specify which panel is the button's parent. For example, the button can be placed in a report header or footer or in the body panel.

page

(Optional) An integer representing the page number of the view on which to place the button.

Return values

Value	Description
Button	Returns a new button.

Approach: New method (ChartView class)

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

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

Creates a new ChartView object.

Syntax

Set *chartviewname* = **New ChartView**(*parent, xaxis, yaxis, series, calculation* |[*charttype*], [*maintable*]*)*

Parameters

parent

A document representing the name of the parent document the chart is based on.

xaxis

A string representing the x-axis field.

yaxis

A string representing the y-axis fields.

series

A string representing the field for the series.

calculation

A long representing the type of calculation. The calculation types are provided as enumerators.

charttype

(Optional) A long representing the chart type. The chart types are provided as enumerators.

maintable

(Optional) A string representing the main table the chart is based on.

Return values

Value	Description
ChartView	Returns a new chart.

Approach: New method (CheckBox class)

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

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

Creates a new checkbox.

Syntax

Set *checkboxname* = **New CheckBox** (*parent,* [*page*])

Parameters

parent

An object from the Panel class representing where the CheckBox is placed. Use to specify which panel is the checkbox's parent. For example, a checkbox can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view to place the CheckBox on.

Return values

Value	Description
CheckBox	Returns a new checkbox.

Approach: New method (Collection class)

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

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

Creates a new collection.

Syntax

Set *collectionname* = **New Collection**()

Parameters

None

Return values

Value	Description
Collection	Returns a new collection.

Approach: New method (Color class)

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

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

Creates a new Color object with the red, green and blue values that you specify.

Syntax

Set *colorname* = **New Color**([*red*], [*green*], [*blue*], [*transparent*])

Parameters

red

(Optional) An integer representing the amount of red present in the new color. Values range from 0 to 255. The default value is 0, representing no red.

green

(Optional) An integer representing the amount of green present in the new color. Values range from 0 to 255. The default value is 0, representing no green.

blue

(Optional) An integer representing the amount of blue present in the new color. Values range from 0 to 255. The default value is 0, representing no blue.

transparent

(Optional) An integer representing the amount of transparent color. Values range from 0 to 255. The default value is 255, representing transparent color.

Return values

Value	Description
Color	A new color object with the specified red, green, and blue values.

Usage

If no values are specified, the values are set to 0 and the Color object is black.

Approach: New method (Connection class)

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

Creates a new instance of the Connection class.

Syntax

Set *connectionname* = **New Connection**()

Parameters

None

Return values

Value	Description
Connection	Returns a new instance of the Connection class.

Approach: New method (Crosstab class)

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

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

Creates a new Crosstab view object with the rows, columns, labels, and calculations that you specify.

Syntax

Set *crosstabname* = **New Crosstab**(*parent, rowfields, colfields, bodyfields, bodycalc, rowtotallabel, rowtotalcalc, coltotallabel, coltotalcalc, [maintable]*)

Parameters

parent

The document on which the crosstab is based.

rowfields

A string array representing a row field.

colfields

A string array representing a column field.

bodyfields

A string array representing fields to be calculated in the body of the crosstab.

bodycalc

The calculation type for the body of the crosstab. These calculations are provided as enumerators.

rowtotallabel

A string representing the label for a summary row.

rowtotalcalc

The calculation type for a summary row. These calculations are provided as enumerators.

coltotallabel

A string representing the label for a summary column.

coltotalcalc

The calculation type for a summary column. These calculations are provided as enumerators.

maintable

A main table for the crosstab.

Return values

Value	Description
Crosstab	Returns a new crosstab.

Approach: New method (Document class)

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

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

Creates a new document based on a table.

Syntax

document = *applicationobjectname*. **NewDocument**(*name, datasourcename, filename, userid, password*)

Parameters

name

A string representing the name of a table.

datasourcename

A string representing a table's data source.

filename

A string representing a table's file name.

userid

A string representing a table user's identification name.

password

A string representing a table's password.

Return values

Value	Description
Document	Returns a new document based on a table.

Approach: New method (DropdownBox class)

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

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

Creates a new dropdown box.

Syntax

Set *dropdownboxname* = **New DropdownBox**(*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the drop-down box is placed. Use to specify which panel is the drop-down box's parent. For example, the drop-down box can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view on which to place the drop-down box.

Return values

Value	Description
Drop-down box	Returns a new drop-down box.

Approach: New method (Ellipse class)

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

Creates a new ellipse.

Syntax

Set *ellipsename* = **New Ellipse**(*parent* **|** [*pagenum*])

Parameters

parent

An object from the Panel class representing where the ellipse is placed. Use to specify which panel is the ellipse's parent. For example, the ellipse can be placed on a form or in the body panel of a report.

pagenum

(Optional) An integer representing the page number of the view to place the ellipse on.

Return values

Value	Description
Ellipse	Returns a new ellipse.

Approach: New method (FieldBox class)

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

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

Creates a new field box.

Syntax

Set *fieldboxname* = **New FieldBox**(*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the field box is placed. Use to specify which panel is the field box's parent. For example, the field box can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view to place the field box on.

Return values

Value	Description
Field Box	Returns a new field box.

Approach: New method (Form class)

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

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

Creates a new form.

Syntax

Set *formname* = **New Form**(*document, [maintable]*)

Parameters

document

A document object to put the form on.

maintable

(Optional) A main table for the form.

Return values

Value	Description
Form	Returns a new form.

Approach: New method (LineObject class)

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

Creates a new line.

Syntax

Set *lineobjectname* = **New LineObject**(*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the line object is placed. Use to specify which panel is the line object's parent. For example, the line object can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view on which to place the LineObject.

Return values

Value	Description
LineObject	Returns a new line object.

Approach: New method (ListBox class)

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

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

Creates a new list box.

Syntax

Set *listboxname* = **New ListBox**(*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the line box is placed. Use to specify which panel is the list box's parent. For example, the list box can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view on which to place the list box.

Return values

Value	Description
ListBox	Returns a new list box.

Approach: New method (Picture class)

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

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

Creates a new picture.

Syntax

Set *picturename* = **New Picture**(*parent*, *filename*, [*page*])

Parameters

parent

An object from the Panel class representing where the picture is placed. Use to specify which panel is the picture's parent. For example, the picture can be placed on a form or in the body panel of a report.

filename

A string representing the path and filename of the picture to display.

page

(Optional) An integer representing the page number of the view on which to place the picture.

Return values

Value	Description
Picture	Returns a new picture.

Approach: New method (PicturePlus class)

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

Creates a new PicturePlus field.

Syntax

Set *pictureplusname* = **New PicturePlus**(*parent*, *datatable*, *datafield*, [*page*])

Parameters

parent

An object from the Panel class representing where the PicturePlus field is placed. Use to specify which panel is the PicturePlus's parent. For example, the PicturePlus field can be placed on a form or in the body panel of a report.

datatable

A string representing the data table.

datafield

A string representing the data field.

page

(Optional) An integer representing the page number of the view on which to place the PicturePlus field.

Return values

Value	Description
PicturePlus	Returns a new PicturePlus field.

Approach: New method (Query class)

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

Creates a new query.

Syntax

queryobjectname.**NewQuery**

set queryobjectname = **NewQuery**()

Parameters

None

Return values

Value	Description
Query	Returns a new instance of the query class.

Approach: New method (RadioButton class)

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

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

Creates a new radio button.

Syntax

Set *radiobuttonname* = **New RadioButton**(*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the RadioButton is placed. Use to specify which panel is the radio button's parent. For example, the radio button can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view on which to place the radio button.

Return values

Value	Description
Radio Button	Returns a new radio button.

Approach: New method (Rectangle class)

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

Creates a new rectangle.

Syntax

Set *rectanglename* = **New Rectangle**(*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the rectangle is placed. Use to specify which panel is the rectangle's parent. For example, the rectangle can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view on which to place the Rectangle.

Return values

Value	Description
Rectangle	Returns a new rectangle.

Approach: New method (RepeatingPanel class)

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

Creates a new repeating panel.

Syntax

Set *repeatingpanel* = **New** RepeatingPanel(*parent, tablename* | [*pagenum*])

Parameters

parent

An object from the Panel class representing where the repeating panel is placed. Use to specify which panel is the repeating panel's parent. For example, the repeating panel can be placed on a form or in the body panel of a report.

tablename

A string representing the table the view is based on.

pagenum

(Optional) An integer representing the page number of the view on which to place the repeating panel.

Return values

Value	Description
Repeating Panel	Returns a new repeating panel.

Approach: New method (Report class)

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

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

Creates a new report.

Syntax

Set *reportname* = **New Report**(*parent,* [*maintable*])

Parameters

parent

An object from the Document class representing where the report is placed. Use to specify which document is the report's parent.

maintable

(Optional) A string representing the table the view is based on.

Return values

Value	Description
Report	Returns a new report.

Approach: New method (ResultSet class)

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

Creates a new ResultSet.

Syntax

resultsetname. **New ResultSet**

Set *resultsetname* = **New ResultSet**()

Parameters

None

Return values

Value	Description
ResultSet	Returns a new result set.

Approach: New method (RoundRect class)

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

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

Creates a new round rectangle object.

Syntax

Set *roundrectanglename* = **New** RoundRect(*parent*, [*page*])

Parameters

parent

An object from the Panel class representing where the rounded rectangle is placed. Use to specify which panel is the rounded rectangle's parent. For example, the rounded rectangle can be placed on a form or in the body panel of a report.

page

(Optional) An integer representing the page number of the view on which to place the rounded rectangle object.

Return values

Value	Description
Round rectangle	Returns a new rounded rectangle.

Approach: New method (SummaryPanel class)

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

Creates a new summary panel.

Syntax

Set *summarypanelname* = **New SummaryPanel**(*parent*)

Parameters

parent

An object from the BodyPanel class representing a summary panel for a report. Use to add a summary panel to a report.

Return values

Value	Description
SummaryPanel	Returns a new summary panel.

Approach: New method (TextBox class)

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

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

Creates a new textbox .

Syntax

Set *textboxname* = **New TextBox**(*parent*, [*text*], [*page*])

Parameters

parent

An object from the Panel class representing where the text box is placed. Use to specify which panel is the text box's parent. For example, the text box can be placed on a form or in the body panel of a report.

text

A string representing text to be displayed in the text box.

page

(Optional) An integer representing the page number of the view on which to place the text box.

Return values

Value	Description
Text box	Returns a new text box object.

Approach: New method (Worksheet class)

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

Creates a new worksheet.

Syntax

Set *worksheetname* **= New Worksheet**(*parent,* [*maintable*])

Parameters

parent

An object from the Document class representing where the worksheet is placed. Use to specify which document is the worksheet's parent.

maintable

A string representing the table for the new worksheet.

Return values

Value	Description
Worksheet	Returns a new worksheet.

Approach: NewPage method

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

Adds a new page to a form.

Syntax

formname.**NewPage**

value = *formname*.**NewPage**()

Parameters

None

Return values

Value	Description
TRUE	A new page was added to the form.
FALSE	A new page was not added to the form.

Approach: NextRecord method

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

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

Makes the next record in the found set the current record. Using this LotusScript command is the same as clicking the NextRecord SmartIcon. If the current record is the last record, nothing happens.

Syntax

docwindowobjectname.**NextRecord**

Parameters

None

Return values

None

Approach: NextRow method

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

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

Moves the pointer to the next row in a result set. You get an error message if there is no result set, or you are at the last row in a result set.

Syntax

resultsetobjectname.**NextRow**

integer = *resultsetobjectname*.**NextRow**()

Parameters

None

Return values

Value	Description
TRUE	The next row was reached.
FALSE	The next row was not reached.

Approach: NumColumns method

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

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

Returns the number of columns in the current result set.

Syntax

numbercols = *resultsetobjectname*.**NumColumns**()

Parameters

None

Return values

Value	Description
Number of columns	An integer representing the number of columns in the result set.

Approach: NumParameters method

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

Returns the number of parameters in an SQL statement. The value is 0 if there are no parameters in the last SQL statement.

Syntax

value = *resultsetobjectname*.**NumParameters**()

Parameters

None

Return values

Value	Description
Number of parameters	An integer representing the number of parameters in an SQL statement.

Approach: NumRows method

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

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

Returns the number of rows in the current result set.

Syntax

numrows = *resultsetobjectname*.**NumRows**()

Parameters

None

Return values

Value	Description
NUMROWS	A long representing the number of rows in the result set.

.

Approach: OpenDocument method

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

Opens the specified document.

Syntax

document = *applicationobjectname*.**OpenDocument**(*filename, path*, | [*filetype, password, openro,makevisible*])

Parameters

filename

A string representing the filename of a document.

path

A variant representing the path of a filename.

filetype

(Optional) A string representing the filetype of a document.

password

(Optional) A string representing the password of a document.

openro

(Optional) An integer representing the read-only status of the document.

makevisible

(Optional) An integer representing whether or not the document is made current.

Return values

Value	Description
Document	Returns the specified document.

Approach: Options method

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

Selects the behavior of the ResultSet object.

Syntax

Integer = *resultsetobjectname*.**Options**(*option*)

Parameters

option

An integer representing the option of the ResultSet object. The options are:

- DB_OPTIMISTIC allows for all users to save their changes to a record.
- DB_OVERRIDE is used in response to an update. Requires cursor support from the database.
- DB_PESSIMISTIC allows only the first user to save changes to a record.

Return values

None

Approach: PrevRecord method

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

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

Makes the previous record in the found set the current record. Using this LotusScript command is the same as clicking the PreviousRecord SmartIcon. If the current record is the first record, nothing happens.

Syntax

docwindowobjectname.**PrevRecord**

Parameters

None

Return values

None

Approach: PrevRow method

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

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

Positions the current row pointer to the previous row in a found set. A FALSE is returned if the previous row is not accessible. This may happen when

- You are already at the first row
- There is no valid table to scroll through
- A result set is not cached
- The cache is limited and the ODBC driver does not support cursor or result navigation

Syntax

integer = *resultsetobjectname*.**PreviousRow**

Parameters

None

Return values

Value	Description
TRUE	The current row pointer was set to the previous row.
FALSE	The current row pointer was not set to the previous row.

Approach: Quit method

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

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

Closes and quits Approach.

Syntax

Set *applicationobjectname* = **Quit**(*savechanges*)

Parameters

savechanges

A variant representing whether or not to save changes made in Approach.

Return values

None

Approach: Refresh method

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Repaints an object to refresh the record data.

Syntax

objectname.**Refresh**

integer = *objectname*.**Refresh**()

Parameters

None

Return values

Value	Description
TRUE	The object was updated.
FALSE	The object was not updated.

Approach: Remove method

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

Removes an object in the collection at the specified position.

Syntax

object = *collectionobjectname*.**Remove**(*position*)

Parameters

position

An integer representing a specified position in the collection.

Return values

Value	Description
Object	Returns the object that was removed from the collection.

Approach: RemoveColumn method

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

Removes the specified column from the worksheet. If no column is specified, the current column is removed.

Syntax

integer = *worksheetobjectname*.**RemoveColumn**([*column*])

Parameters

column

(Optional) A string representing the name of a column.

Return values

Value	Description
TRUE	The specified column was removed from the worksheet.
FALSE	The specified column was not removed from the worksheet.

Approach: Repaint method

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

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

Repaints the document window.

Syntax

documentname.**Repaint**

integer = *documentname*.**Repaint**()

Parameters

None

Return values

Value	Description
TRUE	The document window was repainted.
FALSE	The document window was not repainted.

Approach: ReplaceWithResultSet method

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

This method replaces the table for the .APR file with the table being used to create the ResultSet. You see the result set currently in the ResultSet object.

Syntax

table = tableobjectname.**ReplaceWithResultSet**(*resultset, mappedfields*)

Parameters

resultset

A result set representing the data that replaces the original table in the .APR.

mappedfields

An array representing the table for the .APR file being replaced.

Return values

Value	Description
Table	A table used to create the result set.

Approach: Restore method

{button ,AL(`H_APPLICATIONWINDOW_CLASS;H_DOCWINDOW_CLASS;H_WINDOW_CLASS;',0)} See list of classes

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

Restores the window. Using this LotusScript command is the same as restoring the system icon.

Syntax

windowobjectname.**Restore**

Parameters

None

Return values

None

RunProcedure method

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

Executes a stored procedure.

Syntax

integer = *connectionobjectname*.**RunProcedure**(*procedurename, arg1,..., argn*)

Parameters

procedurename

A string representing the name of the procedure to be executed.

arg1

A string representing the first argument of the procedure.

arg

A string representing the last argument of the procedure.

Return values

Value	Description
TRUE	The procedure was executed.
FALSE	The procedure was not executed.

Usage

Not all data sources support or maintain data procedures. Executing a stored procedure standardizes the process of maintaining data and data integrity, at a back-end. Examples of stored procedures include procedures named "Add Employee" and "Credit Report."

Approach: SameColor method

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

Compares RGB values to see if they are the same. You can also compare colors from different products.

Syntax

value = *objcolor*.**SameColor**(*objcolor2*)

Parameters

objcolor2

A color to be evaluated against this color.

Return values

Value	Description
TRUE	The stored RGB values are the same.
FALSE	The stored RGB values are not the same.

Approach: SendToBack method

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Sends a specified display object to the back of the panel, or the bottom of the layer order. For example, you can have a circle around a group of radio buttons. However, to click the buttons, the Circle object has to be in back.

Syntax

objectname.**SendToBack**

value = *objectname*.**SendToBack**()

Parameters

None

Return values

Value	Description
TRUE	The specified display object was sent to the back of the panel.
FALSE	The specified display object was not sent to the back of the panel.

Approach: SetAt method

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

Replaces the object at Position with the second argument. Signals a bounds error if position is less than one, or greater than Count (which indicates that the size of the collection has not changed). SetAt is normally accessed via indexing.

Syntax

variant = *collectionobjectname*.**SetAt**(*position, object*)

Parameters

position

An integer representing an object in the collection.

object

A variant representing the value used to replace an object in the collection.

Return values

Value	Description
Variant	Returns the object previously at position in the collection.

Approach: SetCellFocus method

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

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

Sets the focus to the specified cell.

Syntax

integer = *worksheetobjectname*.**SetCellFocus**(*column*)

Parameters

column

A string representing the name of a column.

Return values

Value	Description
TRUE	The focus was set to the specified cell.
FALSE	The focus was not set to the specified cell.

Approach: SetFieldList method

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

Sets the list items to be displayed in a drop-down box.

Syntax

integer = *dropdownboxname*.**SetFieldList**(*fielddata*, [*fielddescription*], [*fieldfilterfrom*],[*fieldfilterto*])

Parameters

fielddata

A string representing the name of the field in which the list items are stored.

fielddescription

(Optional) A string representing the name of the field used as the description field. If specified, this field is displayed to the user instead of the data in FieldData.

fieldfilterfrom

(Optional) A string representing the field the items are filtered from.

fieldfilterto

(Optional) A string representing the field the items are filtered to.

Return values

Value	Description
TRUE	The field list was set for the object.
FALSE	The field list was not set for the object.

Approach: SetFocus method

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

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

Gives the specified object the focus, as if a user tabbed into or clicked the object. Clicking sets the focus for field boxes, list boxes, drop-down boxes, check boxes, or radio buttons.

Syntax

objectname.**SetFocus**

integer = *objectname*.**SetFocus**()

Parameters

None

Return values

None

Approach: SetList method

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

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

Sets the list items in the drop-down box or field box & list to the data in a specified array.

Syntax

integer = *dropdownboxname*.**SetList**(*stringarray)*

Parameters

stringarray

An array of strings used to set the list items in the drop-down box or field box & list.

Return values

Value	Description
TRUE	The list items are set for the drop-down box or field box & list.
FALSE	The list items are not set for the drop-down box or field box & list.

Approach: SetParameter method

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

Sets the replacement value for a parameter.

Syntax

value = *queryobjectname*.**SetParameter**(*parametername* | [*ordinalvalue*] *value*)

Parameters

parametername

A string representing the name of the parameter to get. You can also use the ordinal value to get a parameter.

ordinalvalue

An integer representing the ordinal value of the parameter to get. Use as an alternate way to point to a parameter, if the name is unknown. You cannot have an ordinal value and a value as parameters at the same time.

value

A string representing the value to set the parameter to. You cannot have a value and an ordinal value as parameters at the same time.

Return values

Value	Description
TRUE	The replacement value for the parameter was set.
FALSE	The replacement value for the parameter was not set.

Usage

A failed status (or FALSE) is returned when the parameter name or ordinal value is not found or is invalid.

Parameters are set in the SQL statement. Unlike ODBC and standard SQL, parameters can actually appear anywhere in an SQL statement. In SQL , a parameter reference is a name surrounded by question marks (?). For example, the name of the parameter in the following example:

Select * from tablename where amount > ?break?

Parameter arguments for string fields must contain single quotation marks, such as 'value'. Otherwise, the SQL statement will fail.

Approach: SetPicture method

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

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

Determines the picture to be displayed in the picture field.

Syntax

integer = *pictureobjectname*.**SetPicture**(*picturefile)*

Parameters

picturefile

A string representing the location and filename for the picture to place in the picture field.

Return values

Value	Description
TRUE	The specified picture was placed in the picture field.
FALSE	The specified picture was not placed in the picture field.

Approach: SetRGB method

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

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

Sets the red, green, or blue (RGB) value for the color object. The RGB values are provided as enumerators.

Syntax

integer = *colorobjectname*.**SetRGB**(*rgbvalue*)

Parameters

rgbvalue

A long representing an RGB value for a color object.

Return values

Value	Description
TRUE	The RGB was set to a different value.
FALSE	The RGB was not set to a different value.

Approach: SetState method

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

Sets the current check box or radio button to a specified value.

Syntax

integer = *objectname*.**SetState**(*value*)

Parameters

value

A short representing the final state of the check box or radio button.

Return values

Value	Description
TRUE	The check box or radio button was set to the value.
FALSE	The check box or radio button was not set to the value.

Approach: SetText method

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

Sets the text for the current row of the specified column. If no column is specified, the current column is used.

Syntax

integer = *worksheetobjectname*.**SetText**(*text* | [*column*])

Parameters

text

A string representing the text to be entered in the current row of the specified column.

column

(Optional) A string representing the name of a column.

Return values

Value	Description
TRUE	The text for the current row of the specified column was set.
FALSE	The text for the current row of the specified column was not set.

Approach: SetValue method

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

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

Sets the current row to a specified value. The data is automatically converted to the native data type.

Syntax

integer = *resultsetobjectname*.**SetValue**(*columnname* | [*columnid*], *value*)

Parameters

columnname

A string representing the name of a column.

columnid

(Optional) An integer representing the id value of the column. Use as an alternate way to specify the column, if the name is unknown.

value

A variant representing the value to which the current row is set.

Return values

Value	Description
TRUE	The current row was set to the new value.
FALSE	The current row was not set to the new value.

Approach: TabNext method

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

Tabs to the next object in the tab list. Using this LotusScript command is the same as pressing the TAB key.

Syntax

docwindowobjectname.**TabNext**

Parameters

None

Return values

None

Approach: TabPrev method

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

Tabs to the previous object in the tab list. Using this LotusScript command is the same as pressing the SHIFT+TAB keys in combination.

Syntax

docwindowobjectname.**TabPrev**

Parameters

None

Return values

None

Approach: Tile method

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

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

Tiles all the DocWindow objects. Using this LotusScript command is the same as if you decided to tile the system icon.

Syntax

*applicationwindowobjectname***.Tile**

Parameters

None

Return values

None

Approach: UpdateRow method

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

Updates a row in a database with the current row, if changed. UpdateRow, like DeleteRow, may not be possible if the found set or database is read-only.

Syntax

resultsetobjectname.**UpdateRow**

integer = *resultsetobjectname*.**UpdateRow**()

Parameters

None

Return values

Value	Description
TRUE	The current row in the database was updated.
FALSE	The current row in the database was not updated.

Approach: Broadcast event

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

Syntax

Call Broadcast(**source**, *parameter*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

parameter

Approach: CellDataChange event

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

Occurs when you change the data in a worksheet cell.

Syntax

Call CellDataChange(**source**, *columnlabel*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

columnlabel

A string representing the label for a worksheet column.

Approach: CellGetFocus event

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

Occurs when a worksheet cell obtains focus by being tabbed into, clicked, or selected by the keyboard.

Syntax

Call CellGetFocus(**source**, *columnlabel*).

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

columnlabel

A string representing the label of the selected worksheet column.

Approach: CellLostFocus event

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

Occurs when a worksheet cell had focus and then losses it by being tabbed out of, or another cell being clicked or selected by the keyboard.

Syntax

Call CellLostFocus(**source**, *columnlabel*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

columnlabel

A string representing the label of the selected worksheet column.

Approach: Change event

{button ,AL(`H_CHECKBOX_CLASS;H_DROPDOWNBOX_CLASS;H_FIELDBOX_CLASS;H_LISTBOX_CLASS;H_RADIOBUTTON_CLASS;',0)} See list of classes

Occurs when the status of a data entry object changes and the change is entered into the database. For example, if the user makes a change in a checkbox selection, this event is triggered when the user clicks on another field, moves to another record, or saves the record.

Syntax

Call Change(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: Click event

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Occurs when an object is selected using the mouse, the keyboard, or the shortcut keys for the object.

Syntax

Call Click(**source***, x, y, flags*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

x

A long which passes the horizontal coordinate, in twips (1,440 twips = 1 inch), of the click location to the subroutine.

y

A long which passes the vertical coordinate, in twips (1,440 twips = 1 inch), of the click location to the subroutine.

flags

A long representing which mouse button was clicked, the right or left.

Usage

Use when you want to initiate an action. For example, you can write a script to change the font color of a paragraph of text when the user clicks a command button.

Approach: CloseWindow event

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

Occurs when the window is closed.

Syntax

Call CloseWindow(**source**, *document*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

document

The document object.

Approach: DocumentClose event

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

Occurs when you close the current open document window.

Syntax

Call DocumentClosed(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: DocumentOpened event

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

Occurs when you open an existing document.

Syntax

Call DocumentOpened(**source**, *document*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

document

The document object.

Approach: DocumentCreated event

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

Occurs when a new document is created in the application.

Syntax

Call DocumentCreated(**source**, *document*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

document

The document object.

Approach: DoubleClick event

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Occurs when the object is double-clicked using the mouse.

Syntax

Call DoubleClick(**source**, *x*, *y*, *flags*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

x

A long which passes the horizontal coordinate, in twips (1,440 twips = 1 inch), of the double-click location to the subroutine.

y

A long which passes the vertical coordinate, in twips (1,440 twips = 1 inch), of the double-click location to the subroutine.

flags

A long representing which mouse button was clicked, the right or left.

Usage

Use when you want to initiate an action. For example, you can write a script to auto-size a rectangle when the user double-clicks it.

Approach: GotFocus event

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Occurs when an object gets focus by being tabbed into, clicked, or selected using the keyboard. An object which has the focus can appear bold, highlighted with a dashed border or highligetd with a dark border.

Syntax

Call GotFocus(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: KeyDown event

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

Occurs when a key is pressed down.

Syntax

Call KeyDown(**source**, *keycode, repeats, flags, overridedefault*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

keycode

An integer passing the key code of the key that was pressed down, such as KEY_F1 or KEY_ALT, to the subroutine. Use the ASCII key codes to determine which key is pressed.

repeats

An integer representing the number of times a keystroke was repeated as the result of holding the key down.

flags

A short representing the bit values for extended keys. For a list of valid flag values, see the table below.

Value	Description
0-7	Scan code.
	Value depends on the original equipment manufacturer.
8	Extended key.
	For example, the right hand ALT and CTRL keys of an enhanced 101- or 102-key keyboard.
	1 = extended key.
	0 = not an extended key.
9-12	Reserved. Do not use.
13	Context code.
	1 = user holds down ALT while pressing the key.
	0 = user presses the key without holding down ALT.
14	Previous key state.
	1 = key is pressed down before a message is sent.
	0 = key is released before message is sent.
15	Transition state.
	1 = key is being released.
	0 = key is being pressed.

overridedefault

An integer representing whether the keystroke behaves according to its default setting or to another setting.

Usage

Use when you want the user to initiate an action. For example, you can write a script that would be executed when the user presses a certain key down.

Approach: KeyPress event

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

Occurs when a key is pressed and released.

Syntax

Call KeyPress(**source**, *keycode, repeats, flags, overridedefault*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

keycode

An integer passing the ASCII keycode of the key that was pressed to the subroutine. Use the ASCII key codes to determine which key is pressed.

repeats

An integer representing the number of times a keystroke was repeated as the result of holding the key down.

flags

A short representing the bit value for extended keys. For a list of valid flag values, see the table below.

Value	Description
0-7	Scan code.
	Value depends on the original equipment manufacturer.
8	Extended key.
	For example, the right hand ALT and CTRL keys of an enhanced 101- or 102-key keyboard.
	1 = extended key.
	0 = not an extended key.
9-12	Reserved. Do not use.
13	Context code.
	1 = user holds down ALT while pressing the key.
	0 = user presses the key without holding down ALT.
14	Previous key state.
	1 = key is pressed down before a message is sent.
	0 = key is released before message is sent.
15	Transition state.
	1 = key is being released.
	0 = key is being pressed.

overridedefault

An integer representing whether the keystroke behaves according to its default setting or to another setting.

Usage

Use when you want the user to initiate an action. For example, you can write a script that would be executed when the user presses a certain key.

Approach: KeyUp event

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

Occurs when a key is released.

Syntax

Call KeyUp (**source**, *keycode, repeats, flags, overridedefault*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

keycode

An integer passing the key code of the key that was pressed down, such as KEY_F1 or KEY_ALT, to the subroutine. Use the ASCII key codes to determine which key is released.

repeats

An integer representing the number of times a keystroke was repeated as the result of holding the key down.

flags

A short representing the bit value for extended keys. For a list of valid flag values, see the table below.

Value	Description
0-7	Scan code. Value depends on the original equipment manufacturer.
8	Extended key. For example, the right hand ALT and CTRL keys of an enhanced 101- or 102-key keyboard. 1 = extended key. 0 = not an extended key.
9-12	Reserved. Do not use.
13	Context code. 1 = user holds down ALT while pressing the key. 0 = user presses the key without holding down ALT.
14	Previous key state. 1 = key is pressed down before a message is sent. 0 = key is released before message is sent.
15	Transition state. 1 = key is being released. 0 = key is being pressed.

overridedefault

An integer representing whether the keystroke behaves according to its default setting or to another setting.

Usage

Use when you want the user to initiate an action. For example, you can write a script that would be executed when the user releases a certain key.

Approach: LostFocus event

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Occurs when an object has focus and then loses it by being tabbed out of, or when another object is clicked or selected using the keyboard.

Syntax

Call LostFocus(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: MailCheck event

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

Occurs when the mail is checked from within Approach.

Syntax

Call MailCheck(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: MailReceived event

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

Occurs when mail is received.

Syntax

Call MailReceived(source)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: MailSend event

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

Occurs when mail is sent.

Syntax

Call MailSend(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: MouseDown event

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_DROPDOWNBOX_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Occurs when a mouse button is pressed down while the mouse pointer is over an object.

Syntax

Call MouseDown(**source**, *x*, *y, flags)*

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

x

A long which passes the horizontal coordinate, in twips (1,440 twips = 1 inch), of the mouse-down location to the subroutine.

y

A long which passes the vertical coordinate, in twips (1,440 twips = 1 inch), of the mouse down location to the subroutine.

flags

A long representing which mouse button was clicked, the right or left.

Approach: MouseOver event

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Occurs when the mouse pointer is over an object.

Syntax

Call MouseOver(**source**, *x*, *y, flags*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

x

A long which passes the horizontal coordinate, in twips (1,440 twips = 1 inch), of the mouse location to the subroutine.

y

A long which passes the vertical coordinate, in twips (1,440 twips = 1 inch), of the mouse location to the subroutine.

flags

A long representing which mouse button was clicked, the right or left.

Approach: MouseUp event

{button ,AL(`H_BUTTON_CLASS;H_CHECKBOX_CLASS;H_DISPLAY_CLASS;H_ELLIPSE_CLASS;H_FIELDBOX_CLASS;H_LINEOBJECT_CLASS;H_LISTBOX_CLASS;H_OLEOBJECT_CLASS;H_PICTUREPLUS_CLASS;H_PICTURE_CLASS;H_RADIOBUTTON_CLASS;H_RECTANGLE_CLASS;H_ROUNDRECT_CLASS;H_TEXTBOX_CLASS;',0)} See list of classes

Occurs when a mouse button is released while the mouse pointer is over an object.

Syntax

Call MouseUp(**source**, *x*, *y, flags*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

x

A long which passes the horizontal coordinate, in twips (1,440 twips = 1 inch), of the mouse up location to the subroutine.

y

A long which passes the vertical coordinate, in twips (1,440 twips = 1 inch), of the mouse up location to the subroutine.

flags

A long representing which mouse button was clicked, the right or left.

Approach: NewRecord event

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

Occurs when a new record is created.

Syntax

Call NewRecord(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: OpenWindow event

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

Occurs when a document window opens.

Syntax

Call OpenWindow(**source**, *document*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

document

The document object.

Approach: Quit event

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

Occurs when Approach is exited.

Syntax

Call Quit(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: RecordChange event

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

Occurs when you change from one record to another.

Syntax

Call RecordChange(**source**

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: RecordCommit event

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

Occurs when a record is committed by the user saving the application, moving to another record, or adding a new record.

Syntax

Call RecordCommit(**source**, *tablename*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

tablename

A string representing the name of a table.

Approach: SelectColumn event

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

Occurs when you select a worksheet column by tabbing into it, clicking it, or select it with the keyboard.

Syntax

Call SelectColumn(**source**, *columnlabe*l)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

*columnlabe*l

A string representing the label of the selected worksheet column.

Approach: SwitchFrom event

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABELS_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

Occurs when you switch from one view to another view. Determines the view from which you switched.

Syntax

Call SwitchFrom(**source**, *oldview*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

oldview

A variant representing the view object from which you switched.

Approach: SwitchTo event

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABELS_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

Occurs when you switch to one view from another view. Determines the view to which you switched.

Syntax

Call SwitchTo(**source**, *newview*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

newview

A variant representing the view object to which you switched.

Approach: UserTimer event

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABELS_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

Occurs when the user timer expires. Set the length of the timer with the TimerInterval property. When the timer expires, the UserTimer event starts and the timer resets to the TimerInterval value and counts down again. This continues until you set the TimerInterval to zero.

Syntax

Call UserTimer(**source**)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

Approach: ViewSwitch event

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

Occurs when you switch from one view to another view.

Syntax

Call ViewSwitch(**source**, *oldview*, *newview*)

Parameters

source

A LotusScript keyword representing the object that receives the event. Always use the word "source" as the parameter and not the current object's name.

oldview

The view object from which you switched.

newview

The view object to which you switched.

```
' ActionBarVisible property
Sub CleanScreen()
     'This program performs the same function
     'that the Clean Screen menu item on the Edit
     'menu does.
     CurrentWindow.Redraw=False   'Turn off redraw temporarily
     'Turn off each bar.
     CurrentWindow.ActionBarVisible=False
     CurrentWindow.IconBarVisible=False
     CurrentWindow.StatusBarVisible=False
     CurrentWindow.ViewTabVisible=False
     CurrentWindow.Redraw=True   'Turn it back on
     CurrentWindow.Repaint   'Now repaint the window.
End Sub
```

```
' Activate method
'Activate makes the specified document the active document and bring
'it to the foregound. In this example, we have 2 documents (.APR's),
'Customer and Accounts. The user is currently looking at the Customer
'document, that is, the Customer document is active. This simple global
'function is passed a document. The function will activate the document
'that is passed to it.

Sub MakeDocActive(Doc As Document)
    Application.Doc.Activate
End Sub
```

```
' ActiveDocument property
'This code checks to see if the active document is the Customer document.
'If it isn't, it makes the Customer document active.

'Checks if the active document is Customer.
If (CurrentApplication.ActiveDocument.Name <> "Customer") Then
         'Makes the Customer document active.
    CurrentApplication.Customer.Activate
End If
```

```
' ActiveDocWindow property
'Retrieves the active document window.
Dim DocWin As DocWindow 'Create a DocWindow object.
Set DocWin = CurrentApplication.ActiveDocWindow 'Retrieve the active DocWindow.
```

```
' ActiveView property
'This code example comes from the click event for the button btnToday
'in the Schedule SmartMaster application. You can find it on the Start view.

Sub Click(Source As Button, X As Long, Y As Long)
     Set CurrentWindow.ActiveView = CurrentDocument.Schedule~ Display
     CurrentView.Body.fbxDateDisplay.text = Format$(Now, "m/d/yy")
     clearDisplay
     readBlock CurrentView.Body.fbxDateDisplay.Text
End Sub
```

```
' AddColumn method
'Example 1
'Creates a new worksheet and uses the AddColumn method to add the
'field QTY to the new Worksheet. Arguments to name the column and
'position the column are optional.
'Context for this example was a button click event with the button
'on a form in the current document.

Dim Wrk As WORKSHEET
Set Wrk = New WORKSHEET(CURRENTDOCUMENT)
Set CurrentWindow.ActiveView=Wrk
Wrk.Name="Wrksheet"
Wrk.AddColumn("QTY")

'Example 2
'Adds a new column to the current worksheet just to the left of the
'worksheet column named QTY2 and names the new column - MyCol.
'The new column shows the values for the QTY field.

Dim AddCol As Integer
AddCol=CurrentView.AddColumn("QTY","MyCol","QTY2")
```

```
AddRow method
'This is the modifyRooms global function from the Schedule
'SmartMaster application.

Function ModifyRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim RS As New ResultSet


     If Con.ConnectTo("dBASE IV") Then
          Set Qry.Connection = Con
          Qry.Tablename = CurrentDocument.Tables(0).Path & "rooms.dbf"
          Set RS.Query = Qry
          If (RS.Execute)Then
               If (RS.NumRows) Then
                    RS.FirstRow
                    Do
                         RS.DeleteRow
                    Loop While (RS.NumRows)
               End If
               ModifyRooms = True
               For i = 0 To Ubound(rooms)
                    RS.AddRow
                    RS.SetValue "room", rooms(i)
                    RS.UpdateRow
               Next
          End If
     End If
     Con.Disconnect
End Function
```

```
' Alignment property
'Right align the data in the field.
Source.Alignment = $LtsAlignmentLeft

'Or

'Retrieves the alignment set on the data in the field.
Dim align As Integer
align = Source.Alignment
```

```
' AllowDrawing property
'PicturePlus fields do not allow drawing by default,
'but you can allow drawing for a PicturePlus field.
'ObjPictPlus is the name of the PicturePlus field.
source.ObjPictPlus.AllowDrawing = True
```

```
' Application property
'Retrieves the Application object.
Dim App As application
Set App = CurrentApplication.Application
```

```
' ApplicationWindow property
'Retrieves the ApplicationWindow object.
Dim AppWin as ApplicationWindow
Set AppWin = CurrentApplication.ApplicationWindow
```

```
' Author property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' Background property
'Set the background properties of an object.
Source.Background.Color.SetRGB(COLOR_RED)

'Set the background of one object to another.
SetSource.Region.Background.Color = Source.Status.Background.Color
```

```
' Baseline property
'Baselines do not display the default, but you can set it to display the 'default.
Source.Border.Baseline = False
```

```
' Black property
'This is a read-only property. It lets you find out how much black
'is in the color.

Dim b As Long  'Create a variable.

b = source.background.color.black  'Put the amount of black in the background.
Print b  'Print the amount of black.
```

```
' Blue property
'This is a read-only property. It lets you find out how much blue
'is in the color.

Dim b As Long  'Create a variable.

b = source.background.color.blue  'Put the amount of blue in the background.
Print b  'Print the amount of blue.'
```

```
' Border property
'Set the properties for the border of this object.
Source.Border.Color.SetRGB(COLOR_BLUE)
```

```
' Bottom (Border) property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    t = 1635 & (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb (color_ultramarine)
End Sub
```

```
' BringToFront method
'Company is a fieldbox on the form.
'This code is being triggered from the GotFocus event
'of a fieldbox on the same form.
Source.Company.BringToFront
```

```
Cascade method
'Cascade all of the open documents (.APR's).
CurrentApplication.ApplicationWindow.Cascade
```

```
CheckedValue property
'Checks if a checkbox is checked and displays a messagebox with
'an appropriate message and the value.
If (Source.ObjCheckBox.IsChecked) Then
    Messagebox("The checkbox is checked and it's value is: " &
Source.ObjCheckBox.CheckedValue)
Else
    Messagebox("The checkbox is not checked and it's value is: " &
Source.ObjCheckBox.UnCheckedValue)
End If
```

```
' ClickedValue property
'This code is meant to be called from the event of an object
'other than the radio button object.
'ObjRadio is the name of the radio button object.
Source.ObjRadio.ClickedValue = "VISA"

'Or

'Get the ClickedValue of the radio button.
Dim Val As String
Val = Source.ObjRadio.ClickedValue
```

```
' Close method
Sub Click(Source As Button, X As Long, Y As Long)
    Dim d As String

    d = Source.fbxDate.Text
    CurrentWindow.Close
End Sub
```

```
' Close (ResultSet)
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim C As New Connection
     Dim Q As New Query
     Dim RS As New ResultSet

     Dim tname As String
     tname = currentdocument.tables(0).tablename

     If C.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Q.Connection = C
          Q.Tablename = currentdocument.tables(0).path & tname
               Set RS.Query = Qry
               If (RS.Execute)Then
                    If (RS.numrows) Then
                         RS.firstrow
                         Do
                              RS.deleteRow
                         Loop While (RS.numrows)
                    End If
               End If
               RS.Close
          c.disconnect
     End If
End Sub
```

```
' CloseWindow method
'This function lays the groundwork for an application that runs
'in the background. After opening a document, you can have
'Approach hide.

Dim rval As Integer   'Return Value

'Minimize Approach (optional)
CurrentApplication.ApplicationWindow.Minimize

'Hide Approach from the user
CurrentApplication.Visible=False

'Have Approach open an application that runs automatically.
rval = CurrentApplication.OpenDocument("AutoApp", "C:\LOTUS\APPROACH")

'Run your program.

'Show Approach to the user
CurrentApplication.Visible=True

'Maximize Approach (optional)
'CurrentApplication.ApplicationWindow.Maximize

'Quit Approach
CurrentApplication.CloseWindow
```

```
' Connection property
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim ResSet As New ResultSet

     Dim TName As String
     TName = CurrentDocument.Tables(0).TableName

     If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Qry.Connection = Con
          Qry.TableName = CurrentDocument.Tables(0).Path & TName
               Set ResSet.Query = Qry
               If (ResSet.Execute)Then
                    If (ResSet.NumRows) Then
                         ResSet.FirstRow
                         Do
                              ResSet.DeleteRow
                         Loop While (ResSet.NumRows)
                    End If
               End If
          Con.Disconnect
     End If
```

```
' ConnectTo method
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim ResSet As New ResultSet

     Dim TName As String
     TName = CurrentDocument.Tables(0).TableName

     If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Qry.Connection = Con
          Qry.TableName = CurrentDocument.Tables(0).Path & TName
               Set ResSet.Query = Qry
               If (ResSet.Execute)Then
                    If (ResSet.NumRows) Then
                         ResSet.FirstRow
                         Do
                              ResSet.DeleteRow
                         Loop While (ResSet.NumRows)
                    End If
               End If
          Con.Disconnect
     End If
End Sub
```

```
' Count property
'We need to access the main table through the data object.
'The first step is to find out the name of the main table.
Dim MnTbl As String
Dim NumTbls As Integer
Dim Tbls As Variant
Dim t As Variant

MnTbl = CurrentView.MainTable   'Get the name of the main table.
Set Tbls = CurrentDocument.Tables   'Get the collection of tables.
NumTbls = Tbls.Count   'Find out how many tables there are.
For i = 1 To NumTbls Step 1
    If (Tbls(i-1).TableName = MnTbl) Then   'If we've found the right one.
        'Data access code goes here
        j=1
    End If
Next
```

```
' CreateDate property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' CurrentPageNum property
'Find out what the current page number is for the
'current view (form, report, and so on)
Dim CrntPage As Integer
CrntPage = CurrentApplication.ActiveView.CurrentPageNum

'Or

'Change the current page of the current view (form, report, worksheet, and so on)
CurrentApplication.ActiveView.CurrentPageNum = 2
```

```
' CurrentRecord property
'Find out what the current record is for the active view (form, report, worksheet, and
so on.)
Dim CrntRec As Long
CrntRec = CurrentApplication.ActiveDocWindow.CurrentRecord

'Or

'Go to a specific record by setting the CurrentRecord property.
'Go to record 100.
CurrentApplication.ActiveDocWindow.CurrentRecord = 100
```

```
' CurrentRow property
'This script program demonstrates the basics of using the Approach
'Data Object (ADO). First, connect to the SQL Server. If that
'succeeds, initialize the query object.  When initializing the
'query object you can specify either a tablename or an SQL statement.
'They are mutually exclusive.

Dim Conn As New connection   'Create a connection object.
Dim Qry As New query   'Create a Query object.
Dim RsltSet As New ResultSet   'Create a ResultSet object.
Dim rval, Row, i As Integer
Dim Data As String

'Connect to Microsoft SQL Server
If (Conn.connectto("SQL Server","darryl","darryl","sqlsvr-nt")) Then
'If the connection succeeds, initialize the Query object named Qry.
    Qry.sql = "SELECT darryl.authors.au_lname, darryl.authors.au_fname,
darryl.titles.title, darryl.titles.price FROM darryl.authors, darryl.titleauthor,
darryl.titlesWHERE (darryl.authors.au_id = darryl.titleauthor.au_id And
darryl.titleauthor.title_id = darryl.titles.title_id) ORDER BY
darryl.authors.au_lname"
    Set Qry.Connection = Conn   'Set the connection property of the query object to
use the connection object you created.
    Set RsltSet.Query = Qry   'Set the Query property of the ResultSet object to use
the query object you created.
    If (RsltSet.Execute) Then   'Execute the query.
        NoCols = RsltSet.NumColumns   'If successful, find out how many rows there
are.
        Print "====================="
        Do Until (RsltSet.CurrentRow = 10)   'For each row...
            Print "====================="
            Row = Ltrim(RsltSet.CurrentRow)   'Get rid of the spaces.
            Print "Row: " & Row   'Print the row number.
            For i = 1 To NoCols   'For each column.
'Print the field name and then the data.
                Data = Rtrim(Ltrim(RsltSet.FieldName(i))) & ": " &
Ltrim(RsltSet.GetValue(RsltSet.FieldName(i)))
                Print Data
            Next i
            RsltSet.NextRow   'Go to the next row.
            Print "====================="
        Loop
        Print "====================="
    End If
    Conn.Disconnect   'Disconnect from the server.
End If
```

```
' Cyan property
'This is a read-only property. It lets you find out how much cyan
'is in the color.

Dim b As Long  'Create a variable.

b = source.background.color.cyan  'Put the amount of cyan in the background.
Print b  'Print the amount of cyan.
```

```
' DataField property
'Retrieves the name of the field the object is based on.
Dim Fld As String
Fld = Source.DataField

'Or

'Sets the field the object is based on.
'Note: Make sure the DataTable property contains the proper table first.
Source.DataField = "LAST NAME"
```

```
' DataTable property
'Retrieves the name of the table for the field the object is based on.
Dim Tbl As String
Tbl = Source.DataTable

'Or

'Sets the table for the field the object is based on.
'Customer is a table name.
Source.DataTable = "CUSTOMER"
```

```
' DeleteRow method
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
    Dim Con As New Connection
    Dim Qry As New Query
    Dim ResSet As New ResultSet

    Dim TName As String
    TName = CurrentDocument.Tables(0).TableName

    If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
        Set Qry.Connection = Con
        Qry.TableName = CurrentDocument.Tables(0).Path & TName
            Set ResSet.Query = Qry
            If (ResSet.Execute)Then
                If (ResSet.NumRows) Then
                    ResSet.FirstRow
                    Do
                        ResSet.DeleteRow
                    Loop While (ResSet.NumRows)
                End If
            End If
        Con.Disconnect
    End If
End Sub
```

```
Description property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' Disconnect method
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim C As New Connection
     Dim Q As New Query
     Dim RS As New ResultSet

     Dim tname As String
     tname = currentdocument.tables(0).tablename

     If C.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Q.Connection = C
          Q.Tablename = currentdocument.tables(0).path & tname
               Set RS.Query = Q
               If (RS.Execute)Then
                    If (RS.numrows) Then
                         RS.firstrow
                         Do
                              RS.deleteRow
                         Loop While (RS.numrows)
                    End If
               End If
          c.disconnect
     End If
End Sub
```

```
' Document property
'Each view has a document that is its parent.
'Retrieve the document on the current active doc window.
'This script will print the name of the .APR file that
'contains the current view.

Print CurrentView.Document.Name
```

```
' DoMenuCommand method
'The script programmer has access to all of the menus in Approach.
'This is useful if you want a program that interacts with the user.
'For instance, you can put the user in Find mode.
CurrentApplication.ApplicationWindow.DoMenuCommand(IDM_FIND)
```

```
' DrillDownView property
'Specify which view will be displayed when the user drills-down
'from a crosstab.
CurrentDocument.Crosstab~ 1.DrillDownView = CurrentDocument.Chart~ 1
```

```
' Enabled property
'This code example comes from the listbox lbxRooms in the Room Setup
view in the Schedule SmartMaster sample application.

Sub Click(Source As Listbox, X As Long, Y As Long)
     If source.text <> "" Then
          source.btnRemove.enabled = True
     Else
          source.btnRemove.enabled = False
     End If

End Sub
```

```
' Execute method
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim ResSet As New ResultSet

     Dim TName As String
     TName = CurrentDocument.Tables(0).TableName

     If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Qry.Connection = Con
          Qry.TableName = CurrentDocument.Tables(0).Path & TName
               Set ResSet.Query = Qry
               If (ResSet.Execute)Then
                    If (ResSet.NumRows) Then
                         ResSet.FirstRow
                         Do
                              ResSet.DeleteRow
                         Loop While (ResSet.NumRows)
                    End If
               End If
          Con.Disconnect
     End If
End Sub
```

```
' Expand property
'Sets the field to expand when printing, if needed.
Source.Expand = True

'Or

'Retrieves the current setting of the Expand property for this object.
Dim Expnd As Integer
Expnd = Source.Expand
```

```
' FieldName method
Dim c As New connection
Dim qu As New query
Dim rs As New resultset
tname = currentdocument.tables(0).tablename
qu.Tablename = currentdocument.tables(0).path & tname
Set qu.connection = c
Set rs.query = qu
If c.connectto("dBASE IV") Then
   If rs.execute Then
      Do
         For i = 1 To RS.NUMCOLUMNS
            Print RS.FIELDNAME(i) & " : " & RS.GETVALUE(i)
         Next
      Loop While RS.NEXTROW
   End If
End If
```

```
' FieldNames property
Sub TableReport(TblName As String)
    'This function prints a report to the output screen on the
    'table whose name is passed in.
    Dim Tbl As Table
    Dim i As Variant

    'Get the Customer table as a table object.
    Set Tbl=CurrentDocument.GetTableByName(TblName)
    'Since this might be a SQL or Notes table, let's first
    'find out if we're still connected.
    If (Tbl.IsConnection) Then
        'Print the tablename
        Print "Tablename: " & Tbl.Tablename
        Print "FileName: " & Tbl.Filename
        Print "FullName: " & Tbl.FullName
        Print "Path: " & Tbl.Path
        Print "# of Fields: " & Str(Tbl.NumFields)
        Print "# of Records: " & Str(Tbl.NumRecords)
        Print "===================="
        'The array is 0 based, so we need to start at 0
        'and end at NumFields-1.
        For i = 0 To Tbl.NumFields-1 Step 1
            'The FieldNames array holds the names of the fields.
            Print "Field:" & Tbl.FieldNames(i)
            'Print "Formula: " & Tbl.GetFieldFormula(Tbl.FieldNames(i))
            Print "Options: " & Tbl.GetFieldOptions(Tbl.FieldNames(i))
            Print "Size: " & Str(Tbl.GetFieldSize(Tbl.FieldNames(i)))
            Print "Type: " & Str(Tbl.GetFieldType(Tbl.FieldNames(i)))
            Print "===================="
        Next
    End If
End Sub
```

```
' FileName property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' FillField method
'This program fills the EntryDate field in the Customer database
'with Today's date.
    Dim rval As Integer  'Return value
    rval = CurrentWindow.FillField("Customer.EntryDate", Today)
If (rval) Then
   MessageBox("Filled Field successfully.")
Else
   MessageBox("Filled Field failed.")
End If
```

```
' FirstRecord method
'Go to the first record.
CurrentApplication.ActiveDocWindow.FirstRecord
```

```
' FirstRow method
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
    Dim Con As New Connection
    Dim Qry As New Query
    Dim ResSet As New ResultSet

    Dim TName As String
    TName = CurrentDocument.Tables(0).TableName

    If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
        Set Qry.Connection = Con
        Qry.TableName = CurrentDocument.Tables(0).Path & TName
            Set ResSet.Query = Qry
            If (ResSet.Execute)Then
                If (ResSet.NumRows) Then
                    ResSet.FirstRow
                    Do
                        ResSet.DeleteRow
                    Loop While (ResSet.NumRows)
                End If
            End If
        Con.Disconnect
    End If
End Sub
```

```
' FontName property
'Retrieve the font name from the current object.
Dim Fnt As String
Fnt = Source.Font.FontName

'Or

'Set the font in the current object.
Source.Font.FontName = "Arial"
```

```
' FullName property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' GetColorFromRGB method
'Change the background color of the field based on the value.
If (Val(Source.Text) < 0) Then
'Create a color object using the GetColorFromRGB function.  We need to pass
'in a long value representing a color.  There are Approach constants defined
'for many colors that can be passed to the function.
    Set Source.Background.Color = CurrentApplication.GetColorFromRGB(COLOR_RED)
Elseif (Val(Source.Text) = 0) Then
    Set Source.Background.Color = CurrentApplication.GetColorFromRGB(COLOR_BLUE)
Elseif (Val(Source.Text) > 0) Then
    Set Source.Background.Color = CurrentApplication.GetColorFromRGB(COLOR_GREEN)
Else
    Set Source.Background.Color = CurrentApplication.GetColorFromRGB(COLOR_WHITE)
End If
```

```
' GetFieldFormula
Sub TableReport(TblName As String)
    'This function prints a report to the output screen on the
    'table whose name is passed in.
    Dim Tbl As Table
    Dim i As Variant

    'Get the Customer table as a table object.
    Set Tbl=CurrentDocument.GetTableByName(TblName)
    'Since this might be a SQL or Notes table, let's first
    'find out if we're still connected.
    If (Tbl.IsConnection) Then
        'Print the tablename
        Print "Tablename: " & Tbl.Tablename
        Print "FileName: " & Tbl.Filename
        Print "FullName: " & Tbl.FullName
        Print "Path: " & Tbl.Path
        Print "# of Fields: " & Str(Tbl.NumFields)
        Print "# of Records: " & Str(Tbl.NumRecords)
        Print "====================="
        'The array is 0 based, so we need to start at 0
        'and end at NumFields-1.
        For i = 0 To Tbl.NumFields-1 Step 1
            'The FieldNames array holds the names of the fields.
            Print "Field:" & Tbl.FieldNames(i)
            'Print "Formula: " & Tbl.GetFieldFormula(Tbl.FieldNames(i))
            Print "Options: " & Tbl.GetFieldOptions(Tbl.FieldNames(i))
            Print "Size: " & Str(Tbl.GetFieldSize(Tbl.FieldNames(i)))
            Print "Type: " & Str(Tbl.GetFieldType(Tbl.FieldNames(i)))
            Print "====================="
        Next
    End If
End Sub
```

```
' GetFieldOptions method
Sub TableReport(TblName As String)
     'This function prints a report to the output screen on the
     'table whose name is passed in.
     Dim Tbl As Table
     Dim i As Variant

     'Get the Customer table as a table object.
     Set Tbl=CurrentDocument.GetTableByName(dept)
          'Print the tablename
          Print "Tablename: " & Tbl.Tablename
          Print "FileName: " & Tbl.Filename
          Print "FullName: " & Tbl.FullName
          Print "Path: " & Tbl.Path
          Print "# of Fields: " & Str(Tbl.NumFields)
          Print "# of Records: " & Str(Tbl.NumRecords)
          Print "====================="
          'The array is 0 based, so we need to start at 0
          'and end at NumFields-1.
          For i = 0 To Tbl.NumFields-1 Step 1
               'The FieldNames array holds the names of the fields.
               Print "Field:" & Tbl.FieldNames(i)
               'Print "Formula: " & Tbl.GetFieldFormula(Tbl.FieldNames(i))
               Print "Options: " & Tbl.GetFieldOptions(Tbl.FieldNames(i))
               Print "Size: " & Str(Tbl.GetFieldSize(Tbl.FieldNames(i)))
               Print "Type: " & Str(Tbl.GetFieldType(Tbl.FieldNames(i)))
               Print "====================="
          Next
End Sub
```

```
' GetFieldSize method
Sub TableReport(TblName As String)
     'This function prints a report to the output screen on the
     'table whose name is passed in.
     Dim Tbl As Table
     Dim i As Variant

     'Get the Customer table as a table object.
     Set Tbl=CurrentDocument.GetTableByName(dept)
          'Print the tablename
          Print "Tablename: " & Tbl.Tablename
          Print "FileName: " & Tbl.Filename
          Print "FullName: " & Tbl.FullName
          Print "Path: " & Tbl.Path
          Print "# of Fields: " & Str(Tbl.NumFields)
          Print "# of Records: " & Str(Tbl.NumRecords)
          Print "====================="
          'The array is 0 based, so we need to start at 0
          'and end at NumFields-1.
          For i = 0 To Tbl.NumFields-1 Step 1
               'The FieldNames array holds the names of the fields.
               Print "Field:" & Tbl.FieldNames(i)
               'Print "Formula: " & Tbl.GetFieldFormula(Tbl.FieldNames(i))
               Print "Options: " & Tbl.GetFieldOptions(Tbl.FieldNames(i))
               Print "Size: " & Str(Tbl.GetFieldSize(Tbl.FieldNames(i)))
               Print "Type: " & Str(Tbl.GetFieldType(Tbl.FieldNames(i)))
               Print "====================="
          Next
End Sub
```

```
' GetFieldType method
Sub TableReport(TblName As String)
     'This function prints a report to the output screen on the
     'table whose name is passed in.
     Dim Tbl As Table
     Dim i As Variant

     'Get the Customer table as a table object.
     Set Tbl=CurrentDocument.GetTableByName(dept)
          'Print the tablename
          Print "Tablename: " & Tbl.Tablename
          Print "FileName: " & Tbl.Filename
          Print "FullName: " & Tbl.FullName
          Print "Path: " & Tbl.Path
          Print "# of Fields: " & Str(Tbl.NumFields)
          Print "# of Records: " & Str(Tbl.NumRecords)
          Print "====================="
          'The array is 0 based, so we need to start at 0
          'and end at NumFields-1.
          For i = 0 To Tbl.NumFields-1 Step 1
               'The FieldNames array holds the names of the fields.
               Print "Field:" & Tbl.FieldNames(i)
               'Print "Formula: " & Tbl.GetFieldFormula(Tbl.FieldNames(i))
               Print "Options: " & Tbl.GetFieldOptions(Tbl.FieldNames(i))
               Print "Size: " & Str(Tbl.GetFieldSize(Tbl.FieldNames(i)))
               Print "Type: " & Str(Tbl.GetFieldType(Tbl.FieldNames(i)))
               Print "====================="
     End If
End Sub
```

```
'GetHandle method
'The GetHandle function retrieves the Windows handle for the
'DocWindow. This comes in handy when you are making API
'calls to Windows that require the WindowHandle.
Dim hWnd As Long
hWnd= CurrentWindow.GetHandle
'You can now call a Window API requiring a handle to a Window
'and pass it to hWnd.
```

```
' GetTableByName method
Sub TableReport(TblName As String)
     'This function prints a report to the output screen on the
     'table whose name is passed in.
     Dim Tbl As Table
     Dim i As Variant

     'Get the Customer table as a table object.
     Set Tbl=CurrentDocument.GetTableByName(dept)
          'Print the tablename
          Print "Tablename: " & Tbl.Tablename
          Print "FileName: " & Tbl.Filename
          Print "FullName: " & Tbl.FullName
          Print "Path: " & Tbl.Path
          Print "# of Fields: " & Str(Tbl.NumFields)
          Print "# of Records: " & Str(Tbl.NumRecords)
          Print "====================="
          'The array is 0 based, so we need to start at 0
          'and end at NumFields-1.
          For i = 0 To Tbl.NumFields-1 Step 1
               'The FieldNames array holds the names of the fields.
               Print "Field:" & Tbl.FieldNames(i)
               'Print "Formula: " & Tbl.GetFieldFormula(Tbl.FieldNames(i))
               Print "Options: " & Tbl.GetFieldOptions(Tbl.FieldNames(i))
               Print "Size: " & Str(Tbl.GetFieldSize(Tbl.FieldNames(i)))
               Print "Type: " & Str(Tbl.GetFieldType(Tbl.FieldNames(i)))
               Print "====================="
          Next
End Sub
```

```
' GetText method
'Retrieves the value of the current row in the LastName field
'in the current view (which is a WORKSHEET).
Dim Txt As String
Txt = CurrentApplication.ActiveView.GetText("LastName")
```

```
' GetValue method
Dim c As New connection
Dim qu As New query
Dim rs As New resultset
tname = currentdocument.tables(0).tablename
qu.Tablename = currentdocument.tables(0).path & tname
Set qu.connection = c
Set rs.query = qu
If c.connectto("dBASE IV") Then
   If rs.execute Then
      Do
         For i = 1 To RS.NUMCOLUMNS
            Print RS.FIELDNAME(i) & " : " & RS.GETVALUE(i)
         Next
      Loop While RS.NEXTROW
   End If
End If
```

```
' Green property
'This is a read-only property. It lets you find out how much green
'is in the color.

Dim b As Long   'Create a variable.

b = source.background.color.green   'Put the amount of green in the 'background.
Print b   'Print the amount of green.
```

```
' Height property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim h As Integer
    Dim i As Integer

    For i = 0 To Ubound(rooms)
        If rooms(i) = roomName Then
            t = i
            i = Ubound(rooms)
        End If
    Next
    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
    tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' HideMargins property
'Find out if the margins are hidden on the current view.
Dim rval As Integer
rval = CurrentApplication.ActiveView.HideMargins

'Or

'Set the margins to be visible on the current view.
CurrentApplication.ActiveView.HideMargins = False
```

```
' IconBarVisible property
Sub CleanScreen()
     'This little program performs the same function
     'that the Clean Screen menu item on the Edit
     'menu does.
     CurrentWindow.Redraw=False   'Turn off redraw temporarily
     'Turn off each bar.
     CurrentWindow.ActionBarVisible=False
     CurrentWindow.IconBarVisible=False
     CurrentWindow.StatusBarVisible=False
     CurrentWindow.ViewTabVisible=False
     CurrentWindow.Redraw=True   'Turn it back on
     CurrentWindow.Repaint   'Now repaint the window.
End Sub
```

```
' IconSets property
'Find out what icon sets are available.
Dim IcnSets As Variant
IcnSets = CurrentApplication.IconSets
```

```
' InsertAfter method
'Inserts one object after another object.
'Inserts the ObjButton object after the LastName object in the layer order.
'After this code is executed, the ObjButton object displays beneath the 'LastName object. LastName displays on top of ObjButton.
Source.ObjButton.InsertAfter(Source.LastName)
```

```
' IsChecked property
'Checks if a checkbox is checked and displays a messagebox with
'an appropriate message and the value.
If (Source.ObjCheckBox.IsChecked) Then
    Messagebox("The checkbox is checked and it's value is: " &
Source.ObjCheckBox.CheckedValue)
Else
    Messagebox("The checkbox is not checked and it's value is: " &
Source.ObjCheckBox.UnCheckedValue)
End If
```

```
' IsCommandChecked method
'Determines if the menu item is checked.
Dim rval As Integer
'Determines if the View - Show Tab Order menu item is checked.
rval = CurrentApplication.ApplicationWindow.IsCommandChecked(IDM_SHOWTABS)
```

```
' IsCommandEnabled method
'Finds out if the File - Save menu item is enabled.
Dim rval As Integer
rval = CurrentApplication.ApplicationWindow.IsCommandEnabled(IDM_SAVE)
```

```
' IsEmpty method
'In this code example, you create a listbox and fill
'the listbox with the views available.
Dim LstBox As ListBox
Dim aryVws(20) As String
Dim NumVws As Integer

'Check to find out if the current view is a form.
    If (CurrentView.Type = $aprForm) Then
        'First find out if the collection of views is empty.
        If (CurrentDocument.Views.IsEmpty=False) Then
            NumVws = CurrentDocument.Views.Count   'Get the number of views
            For i = 0 To NumVws-1   'Fill the array with the view names.
                aryVws(i) = CurrentDocument.Views(i).Name
            Next
        End If
    Else
        Messagebox "You must be on a form."
    End If
```

```
' Italic property
'Find out if the font in the current field is italicized.
'If so, display a message
If(Source.Font.Italic) Then
   Messagebox("The font is italic.")
Else
   Messagebox("The font is not italic")
End If

'Or

'Set the font to be italic in the current field.
Source.Font.Italic = True
```

```
' Keywords property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' LabelAlignment property
'Sets the label to be aligned to the center of the field.
Source.LabelAlignment = $LtsAlignmentHorizCenter

'Or

'Retrieves the current label alignment setting.
Dim LblAlign As Long
LblAlign = Source.LabelAlignment
```

```
' LabelFont property
'Changes the font of the label for the current object to look like the font
'of the LASTNAME object.

Set Source.LabelFont = Source.LASTNAME.LabelFont

'Or

'Create your own font, set the properties, and use it to set the label font.
Dim Fnt as Font
Set Fnt = Source.Labelfont
Fnt.Bold = True   'Bold the font.
Fnt.FontName = "Arial"   'Set the font to Arial
Fnt.Size = 10   'Set the size to a 10pt font.
```

```
' LabelPosition property
'Displays the label below the object.
Source.LabelPosition = $LtsPositionTop

'Or

'Retrieves the current position of the label for this object.
Dim LblPos As Long
LblPos = Source.LabelPosition
```

```
' LabelText property
'Sets the text for the object's label.
'Set the label to Last Name.
Source.LabelText = "Last Name"

'Or

'Retrieves the current label for the object.
Dim Lbl As String
Lbl = Source.LabelText
```

```
' LastModified property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window.

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' LastRecord method
'Go to the last record.
CurrentApplication.ActiveDocWindow.LastRecord
```

```
' LastRow method
'This code is pulled from the deleteScheduledRemovedRooms global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
    Dim C As New Connection
    Dim Q As New Query
    Dim RS As New ResultSet

    Dim tname As String
    tname = currentdocument.tables(0).tablename

    If C.ConnectTo("dBASE IV") Then   'Connect to dBASE.
        Set Q.Connection = C
        Q.Tablename = currentdocument.tables(0).path & tname
            Set RS.Query = Q
            If (RS.Execute)Then
                If (RS.numrows) Then
                    RS.lastrow   'Go to the last row.
                End If
            End If
        c.disconnect
    End If
End Sub'
```

```
' Left property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim h As Integer
    Dim i As Integer

    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
    tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' Left (Border)property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim h As Integer
    Dim i As Integer

    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
    tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' LeftMargin property
'Find out what the current left margin is (in TWIPS) for the active view.
Dim LeftMrgn As Integer
LeftMrgn = CurrentApplication.ACTIVEVIEW.LeftMargin

'Or

'Set the left margin for the active view to a new value.
CurrentApplication.ACTIVEVIEW.LeftMargin = 1440
```

```
' LineStyle property
'Reference the Pattern of the line using the LineStyle property.
Sub Click(Source As Button, X As Long, Y As Long, Flags As Long)
          Source.ObjLine.LineStyle.Pattern = $ltsLineStyleDot
End Sub
```

```
' MacroClick property
'Set the macro that executes when the object is clicked.
MainMenuButton.MacroClick = "Go To Main Menu"
```

```
' MacroDataChange property
'Sets the macro that executes when the data in the object changes.
'The macro Go To Customer Information is launched when the data in this
'object changes.
Source.MacroDataChange = "Go To Customer Information"

'Or

'Get the name of the macro that is currently set.
Dim mstr As String
mstr = Source.MacroDataChange
```

```
' MacroTabIn property
'Sets the macro that is executed when the object is tabbed into.
'The macro Go To Customer Information is executed when this object is
'tabbed into.
Source.MacroTabIn = "Go To Customer Information"

'Or

'Get the name of the macro that is currently set.
Dim mstr As String
mstr = Source.MacroTabIn

'Or

'Assign the macro to another object.
Source.Field2.MacroTabIn = Source.MacroTabIn
```

```
' MacroTabOut property
'Sets the macro that is executed when the object is tabbed into.
'The macro Go To Customer Information is launched when this object
'is tabbed into.
Source.MacroTabOut = "Go To Customer Information"

'Or

'Get the name of the macro that is currently set.
Dim mstr As String
mstr = Source.MacroTabOut

'Or

'Assign the macro to another object.
Source.Field2.MacroTabOut = Source.MacroTabOut
```

```
' Magenta property
'This is a read-only property.  It lets you find out how much magenta
'is in the color.

Dim b As Long   'Create a variable.

b = source.background.color.magenta
Print b   'Print the amount of magenta.
```

```
' MainTable property
'We need to access the main table through the data object.
'The first step is to find out where the main table is.
Dim MnTbl As String
Dim NumTbls As Integer
Dim Tbls As Variant
Dim t As Variant

MnTbl = CurrentView.MainTable   'Get the name of the main table.
Set Tbls = CurrentDocument.Tables   'Get the collection of tables.
NumTbls = Tbls.Count   'Find out how many tables there are.
For i = 1 To NumTbls Step 1
    If (Tbls(i-1).TableName = MnTbl) Then   'If we've found the right one.
        j=1
    End If
Next
```

```
' MakeNamedStyle method
'Create a named style called MyStyle based on the attributes
'of the current field.
Dim rval As Integer
rval = Source.MakeNamedStyle("MyStyle")
```

```
' Maximize method
'This function lays the groundwork for an application that runs
'in the background. After opening a document, you can have
'Approach hide.

CurrentApplication.ApplicationWindow.Minimize
```

```
' MenuBar property
'Change the menu for the current view to
'a customer menu named EasyMenu.

Dim Mnu As Variant
Dim NumMnus As Integer

Mnu = CurrentApplication.Menus   'Get the list of menus.

NumMnus = Ubound(Mnu)      'Find out how many custom menus there are.
'Arrays are 0 based.
For i = 0 To NumMnus Step 1
     If (Mnu(i) = "EasyMenu") Then   'If we've found the right one.
          CurrentView.MenuBar = Mnu(i)   'Change it.
          i = NumMnus
     Elseif (i = NumMnus) Then   'If the right one isn't there.
          Messagebox "EasyMenu is not available.",,"Menus"
     End If
Next
```

```
' Menus property
'Change the menu for the current view to
'a customer menu named EasyMenu.

Dim Mnu As Variant
Dim NumMnus As Integer

Mnu = CurrentApplication.Menus   'Get the list of menus.

NumMnus = Ubound(Mnu)      'Find out how many custom menus there are.
'Arrays are 0 based.
For i = 0 To NumMnus Step 1
     If (Mnu(i) = "EasyMenu") Then   'If we've found the right one.
          CurrentView.MenuBar = Mnu(i)   'Change it.
          i = NumMnus
     Elseif (i = NumMnus) Then   'If the right one isn't there.
          Messagebox "EasyMenu is not available.",,"Menus"
     End If
Next
```

```
' Minimize method
'This function lays the groundwork for an application that runs
'in the background. After opening a document, you can hide Approach.

'Minimize Approach
CurrentApplication.ApplicationWindow.Minimize
```

```
' Modified property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' Name property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(Txt As String, Start As Double, Finish As Double, RoomName As String)
     Dim TxtBox As textbox

     Dim h As Integer
     Dim i As Integer

     t = 1635 + (330 * t)
     Set TxtBox = New TextBox(CurrentView.Body)
     TxtBox.Text = " " & txt & " "
     TxtBox.Font.Size = 8
     TxtBox.Border.Style = $ltsBorderStyleNone
     TxtBox.Border.Left = True
     TxtBox.Border.Right = True
     TxtBox.Border.Top = False
     TxtBox.Border.Bottom = False
     TxtBox.Border.Width = $apr1point
     TxtBox.Border.Color.SetRGB COLOR_ULTRAMARINE
     TxtBox.Background.Color.SetRGB COLOR_50_GRAY
     TxtBox.Height = 325
     TxtBox.Top = t
     TxtBox.Left = (((Start - 8) * 750) + 960)
     TxtBox.Width =  (750 * (Finish - Start))
     TxtBox.Name = "TxtBox" & Str$(TxtBox.Top) & Str$(TxtBox.Left)
End Sub
```

```
' NamedFindSort property
'Change the named find/sort to the "CA" named find/sort
Dim Finds As Variant
Dim NumFinds As Integer

Finds = CurrentDocument.NamedFindSorts

NumFinds = Ubound(Finds)       'Find out how many named finds there are.
'Arrays are 0 based.
For i = 0 To NumFinds Step 1
     If (Finds(i) = "CA") Then   'If we've found the right one.
          CurrentWindow.NamedFindSort = Finds(i)   'Change it.
     Elseif (i = NumFinds) Then   'If the right one isn't there.
          Messagebox "The CA named find/sort is not available.",,"Find/Sort"
     End If
Next
```

```
' NamedFindSorts property
'Change the named find/sort to the "CA" named find/sort
Dim Finds As Variant
Dim NumFinds As Integer

Finds = CurrentDocument.NamedFindSorts

NumFinds = Ubound(Finds)       'Find out how many named finds there are.
'Arrays are 0 based.
For i = 0 To NumFinds Step 1
     If (Finds(i) = "CA") Then   'If we've found the right one.
          CurrentWindow.NamedFindSort = Finds(i)   'Change it.
     Elseif (i = NumFinds) Then   'If the right one isn't there.
          Messagebox "The CA named find/sort is not available.",,"Find/Sort"
     End If
Next
```

```
' NamedStyle property
'Change the menu for the current view to
'a customer menu named EasyMenu.

Dim Styl As Variant
Dim NumStyls As Integer

Styl = CurrentDocument.NamedStyles   'Get the list of menus.

NumStyls = Ubound(Styl)    'Find out how many named styles there are.
'Arrays are 0 based.
For i = 0 To NumStyls Step 1
     If (Styl(i) = "CompanyStyle") Then   'If we've found the right one.
          Source.NamedStyle = Styl(i)   'Change it.
          i = NumStyls
     Elseif (i = NumStyls) Then   'If the right one isn't there.
          Messagebox "The company named style is not available.",,"Styles"
     End If
Next
```

```
' NamedStyles property
'Change the menu for the current view to
'a customer menu named EasyMenu.

Dim Styl As Variant
Dim NumStyls As Integer

Styl = CurrentDocument.NamedStyles   'Get the list of menus.

NumStyls = Ubound(Styl)    'Find out how many named styles there are.
'Arrays are 0 based.
For i = 0 To NumStyls Step 1
     If (Styl(i) = "CompanyStyle") Then   'If we've found the right one.
          Source.NamedStyle = Styl(i)   'Change it.
          i = NumStyls
     Elseif (i = NumStyls) Then   'If the right one isn't there.
          Messagebox "The company named style is not available.",,"Styles"
     End If
Next
```

```
' New (Button) method
'This code creates a new button.
    Dim Btn As Button
    Set Btn = New Button(CurrentApplication.ActiveView.Body)   'Creates a new button
on the current view on page 1.
    Btn.Left = 2000   'Position the button
    Btn.Top = 2000
    Btn.Width = 1750
    Btn.Height = 640
    Btn.NonPrinting = True   'Do not print this button.
    Btn.Text = "Click Here"   'Set the text to be displayed on the button.
    Btn.NamedStyle = "Default"   'Set the attributes using a pre-existing named style.
```

```
' New (ChartView) method
'Creates a new ChartView.
'In the example below, the SALESREP field is grouped on the X axis;
'the QTY field is summed on the Y axis; and a new series is created
'or each distinct value in 'product CATEGORY field.  The X, Y, and S
'arguments are arrays of strings.
'The enumeration constants - $aprCalcSum and $aprChartTypeBar - signify
'that the values 'represented by the Y argument should be summed and that
'the chart is a bar chart. Precede 'each enumeration constant with a $.
'Once the chart is created the script makes the new chart the active view
'and names it.
'The context for this example might be a button on an existing view.

Dim X(1) As String
Dim Y(1) As String
Dim S(1) As String

X(1)= "SALESREP"
Y(1) = "QTY"
S(1)= "CATEGORY"

Dim CHRT As CHARTVIEW
Set CHRT = New CHARTVIEW(CURRENTDOCUMENT, X, Y, S, $aprCalcSum, $aprChartTypeBar)

Set CURRENTWINDOW.ACTIVEVIEW=CHRT
CHRT.NAME="MyCHRT"
```

```
' New (Checkbox)method
'This code creates a new checkbox.

Set ChkBox = New CheckBox(CurrentApplication.ActiveView.Body)  'Create a new checkbox
on the current view on page 1.
'Now let's setup the checkbox.
ChkBox.Left = 400   'In TWIPS
ChkBox.Top = 400   'In TWIPS
ChkBox.NamedStyle = MyStyle   'Set the attributes using a pre-existing named style.
ChkBox.Labeltext = ""   'Set the label for the checkbox.
ChkBox.DataTable = "Customer"   'Assigns the checkbox to a field in the Customer
table.
ChkBox.IsChecked = True   'Check the checkbox by default.
```

```
' New (Collection) method
'Collections give you the ability to make a "group" of objects.
Dim rval As Integer   'Return value
Dim CustomerColl As Collection

Set CustomerColl = New Collection()

rval = CustomerColl.Add(Source.FirstName)
rval = CustomerColl.Add(Source.LastName)
rval = CustomerColl.Add(Source.Address)
rval = CustomerColl.Add(Source.City)
rval = CustomerColl.Add(Source.State)
rval = CustomerColl.Add(Source.Postal_Cod)
rval = CustomerColl.Add(Source.Company)

'Now copy the Company field to the 3rd object
'of the collection.
rval = CustomerColl.SetAt(3, Source.Company)

'The only fields we need now are the FirstName,
'LastName, and Company fields.  So let's remove
'the rest from the collection.  Since the SetAt function
'copies the object to the specified location, we have
'2 copies of the Company field in the collection.  So
'Let's start at the end of the collection and work our way
'backwards to get rid of the Company field at the end of
'the collection.
'Arrays are 0 based.
i = CustomerColl.Count - 1
Do While i >2
     If (CustomerColl(i).Name = "COMPANY") Then
          CustomerColl.Remove(i+1)
          'Since we've deleted an object, we need to reduce
          'the number of items in the collection by 1.
          i = i - 1
     End If
     If (CustomerColl(i).Name = "ADDRESS") Then
          CustomerColl.Remove(i+1)
          i = i - 1
     End If
     If (CustomerColl(i).Name = "CITY") Then
          CustomerColl.Remove(i+1)
          i = i - 1
     End If
     If (CustomerColl(i).Name = "STATE") Then
          CustomerColl.Remove(i+1)
          i = i - 1
```

```
    End If
    If (CustomerColl(i).Name = "POSTAL_COD") Then
        CustomerColl.Remove(i+1)
        i = i - 1
    End If
Loop
```

```
' New (Color) method
'Creates a new color
Dim MyRed As Color
Set MyRed = New Color(224, 31, 36)

'Set the background color to MyRed.
Source.Background.Color = MyRed
```

```
' New (Crosstab) method
'The example below creates a new crosstab using the example fields:
'SALESREP, CATEGORY, and QTY for the rows, columns and body of the crosstab
'respectively. The arguments represented by R, C, B are arrays of strings.

'Enumeration constants ($aprCalcSum) are used to designate the summary
'calculation type for the body cells, as well as for the row and column
'totals. Each enumeration constant is preceded with a dollar sign. String
'values (TOTAL) are used to label the grand summary row and column of the
'crosstab.
'The arguments for row, column and body are arrays of strings. Thus
'multi-level summaries can be created, as well as adding more than one field
'to the body.

Dim R(1) As String
Dim C(1) As String
Dim B(1) As String

R(1) = "SALESREP"
C(1) = "CATEGORY"
B(1) = "QTY"

Dim CR As Crosstab

Set CR = New CROSSTAB(CURRENTDOCUMENT,R,C,B,$aprCalcSum,"TOTAL",$aprCalcSum,"TOTAL",
$aprCalcSum)
```

```
' New (Document) method
'Creates a new document.
Dim Con As New Connection
Dim Qry As New Query
Dim ResSet As New ResultSet
Dim Doc As Document

If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
     Set Qry.Connection = Con
     Qry.TableName = "C:\LOTUS\APPROACH\CUSTOMER.DBF"
     Set ResSet.Query = Qry
     If (ResSet.Execute)Then
          Set Doc = New Document(ResSet)
     End If
     Con.Disconnect
End If
```

```
' New (DropdownBox) method
'Create a DropdownBox on the current view on page #1.
Sub Click(Source As Button, X As Long, Y As Long, Flags As Long)
    Dim DropDnBox As DropDownBox

    Set DropDnBox = New DropDownBox(Source.Parent,1)
End Sub
```

```
' New (FieldBox) method
'Creates a new fieldbox on the current form on page #1. After you create
'the fieldbox,you can "attach" it to a field in the database by specifying
'the DataTable and DataField properties.
Sub Click(Source As Button, X As Long, Y As Long, Flags As Long)
    Dim FldBox As FieldBox

    Set FldBox = New FieldBox(Source.Parent,1)
    FldBox.DataTable = "Customer"
    FldBox.DataField = "Company"
End Sub
```

```
' New (Form) method
'This code segment creates a new form and adds a field box to the new form.
'The field box is then set to display the values from a field in the
'current table.

Dim Frm As Form
Dim FB As Fieldbox

Set Frm = New Form(CurrentDocument)
Set CurrentWindow.ActiveView=Frm
Frm.Name="MyForm"

Set FB=New Fieldbox (currentdocument.myform.body)
FB.width=1440
FB.height=360
FB.datatable="INVOICE"
FB.datafield="SALESREP"
```

```
' New (Line) method
'Creates a new line on the current view on page #1.
Sub Click(Source As Button, X As Long, Y As Long, Flags As Long)
    Dim L As LineObject

    Set L = New LineObject(Source.Parent,1)
    L.Left = 1440
    L.Top = 1440
End Sub
```

```
' New (ListBox)
'This code example creates a listbox and fills
'the listbox with the available views.
Dim LstBox As ListBox
Dim aryVws(20) As String
Dim NumVws As Integer

'Check to find out if the current view is a form.
If (CurrentView.Type = $aprForm) Then
     'Create a new listbox on the current view
     Set LstBox = New ListBox(CurrentView.Body, 1)
     NumVws = CurrentDocument.Views.Count   'Get the number of views
     For i = 0 To NumVws-1   'Fill the array with the view names.
          aryVws(i) = CurrentDocument.Views(i).Name
     Next
Else
     Messagebox "You must be on a form."
End If
```

```
' New (Picture) method
'This global function was taken from the Viewer
'SmartMaster application.

Function createPictures(f As form) As Integer
    On Error Resume Next

    Dim obj As Integer
    Dim c As collection
    Dim p As picture
    Dim pnl As panel

    Set pnl = f.body
    Set c = f.objectlist
    Forall o In c
        If (o.type = $aprPicture) Then
            Delete o
        End If
    End Forall

    obj = 0
    For i = 1 To ((Ubound(pictureList)/numPicColumns) + 1)
        If i = 1 Then
            t = t + 900 + 270 + 90
        Else
            t = t + 1440 + 270
        End If
        l = 0
        For j = 1 To numPicColumns
            If (j <= (Ubound(pictureList) + 1)/i) Then
                If j = 1 Then
                    l = l + 720 + 270 + 90
                Else
                    l = l + 1440 + 270
                End If
                Set p = New picture(pnl, pictureList(obj))
                p.height = 1440
                p.width = 1440
                p.left = l
                p.top = t
                obj = obj + 1
            End If
        Next
    Next
    runapproachmacro("switchToViewer")
End Function
```

```
' New (RadioButton) method
'Creates a new RadioButton on the current form on page #1. This example
'was written in the click event of a display object (such as a fieldbox).
Sub Click(Source As Button, X As Long, Y As Long, Flags As Long)
    Dim rb As RadioButton
    Set rb = New RadioButton(Source.Parent,1)
End Sub
```

```
' New (Report) method
'Creates a new report and designates INVOICE as the main table for that
'report. Once the new report is created, the script makes the new report the
'active view and names it. Then a summary panel that groups records by
'SALESREP is added to the report and named.
'Once the summary panel is added to the report, the fields SALESREP and
'QtybyRep are added to the summary panel. The body height is set to zero,
'so that only one row appears for each SALESREP with a subtotal of QTY for
'that SALESREP. The QTY subtotal is calculated by adding the previously
'defined summary calculaltion (SSUM of QTY) to the summary panel.

Dim Rpt As Report
Dim Spanel As SummaryPanel
Dim FB As FieldBox
Dim SumFB As FieldBox

Set Rpt = New Report (CurrentDocument, "INVOICE")
Rpt.name="MyReport"
Set Spanel = New SummaryPanel(CurrentDocument.MyReport.body)
Spanel.Height=360
Spanel.background.color.setrgb(COLOR_VANILLA)
Spanel.GROUPBYDATATABLE="INVOICE"
Spanel.GROUPBYDATAFIELD="SALESREP"
Spanel.name="BySalesRep"
Set FB=New Fieldbox (CurrentDocument.MyReport.BySalesRep)
FB.width=1440
FB.height=360
FB.datatable="INVOICE"
FB.datafield="SALESREP"
CurrentDocument.MyReport.Body.Height=0

Set SumFB= New FieldBox(CurrentDocument.MyReport.BySalesRep)
SumFB.width=1440
SumFB.height=360
SumFB.LEFT=2880
SumFB.datatable="INVOICE"
SumFB.datafield="QtybyRep"
```

```
' New (RoundRect) method
   'Create a new RoundRect in the current page on page #1.
   Dim RoundRectangle As RoundRect
   Set RoundRectangle = New RoundRect(Source.Parent,1)
```

```
' New (SummaryPanel) method
'Creates a new report and designates INVOICE as the main table for that
'report. Once the new report is created, the script makes the new report
'the active view and names it. Then a summary panel that groups records by
'SALESREP is added to the report and named.
'Once the summarypanel is added to the report, the fields SALESREP and
'QtybyRep are added to the summary panel. The report body height is set to
'zero, so that only one row appears for each SALESREP with a subtotal of QTY
'for that SALESREP. The QTY subtotal is calculated by adding the previously
'defined summary calculaltion (SSUM of QTY) to the summary panel.

Dim Rpt As Report
Dim Spanel As SummaryPanel
Dim FB As FieldBox
Dim SumFB As FieldBox

Set Rpt = New Report (CurrentDocument, "INVOICE")
Rpt.name="MyReport"
Set Spanel = New SummaryPanel(CurrentDocument.MyReport.body)
Spanel.Height=360
Spanel.background.color.setrgb(COLOR_VANILLA)
Spanel.GROUPBYDATATABLE="INVOICE"
Spanel.GROUPBYDATAFIELD="SALESREP"
Spanel.name="BySalesRep"
Set FB=New Fieldbox (CurrentDocument.MyReport.BySalesRep)
FB.width=1440
FB.height=360
FB.datatable="INVOICE"
FB.datafield="SALESREP"
CurrentDocument.MyReport.Body.Height=0
```

Set SumFB= New FieldBox(CurrentDocument.MyReport.BySalesRep)

```
SumFB.width=1440
SumFB.height=360
SumFB.LEFT=2880
SumFB.datatable="INVOICE"
SumFB.datafield="QtybyRep"
```

```
' New (Textbox) method
Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim h As Integer
    Dim i As Integer

    For i = 0 To Ubound(rooms)
        If rooms(i) = roomName Then
            t = i
            i = Ubound(rooms)
        End If
    Next
    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
End Sub
```

```
' NewPage method
'Adds a new page to the active view.
CurrentApplication.ActiveView.NewPage
```

```
' NextRecord method
'Go to the next record.
CurrentApplication.ActiveDocWindow.NextRecord
```

```
' NextRow method
Dim c As New connection
Dim qu As New query
Dim rs As New resultset
tname = currentdocument.tables(0).tablename
qu.Tablename = currentdocument.tables(0).path & tname
Set qu.connection = c
Set rs.query = qu
If c.connectto("dBASE IV") Then
   If rs.execute Then
      Do
         For i = 1 To RS.NUMCOLUMNS
            Print RS.FIELDNAME(i) & " : " & RS.GETVALUE(i)
         Next
      Loop While RS.NEXTROW
   End If
End If
```

```
' NonPrinting property
'Sets whether the object prints or not.
'Object prints when printing.
Source.NonPrinting = True

'Or

'Get the value of the property.
Dim rval As Integer
rval = Source.NonPrinting
```

```
' NumColumns method
Dim c As New connection
Dim qu As New query
Dim rs As New resultset
tname = currentdocument.tables(0).tablename
qu.Tablename = currentdocument.tables(0).path & tname
Set qu.connection = c
Set rs.query = qu
If c.connectto("dBASE IV") Then
   If rs.execute Then
      Do
         For i = 1 To RS.NUMCOLUMNS
            Print RS.FIELDNAME(i) & " : " & RS.GETVALUE(i)
         Next
      Loop While RS.NEXTROW
   End If
End If
```

```
' NumFields property
Sub TableReport(TblName As String)
    'This function prints a report to the output screen on the
    'table whose name is passed in.
    Dim Tbl As Table
    Dim i As Variant

    'Get the Customer table as a table object.
    Set Tbl=CurrentDocument.GetTableByName(TblName)
    'Since this might be a SQL or Notes table, let's first
    'find out if we're still connected.
    If (Tbl.IsConnection) Then
        'Print the tablename
        Print "Tablename: " & Tbl.Tablename
        Print "FileName: " & Tbl.Filename
        Print "FullName: " & Tbl.FullName
        Print "Path: " & Tbl.Path
        Print "# of Fields: " & Str(Tbl.NumFields)
        Print "# of Records: " & Str(Tbl.NumRecords)
        Print "===================="
        'The array is 0 based, so we need to start at 0
        'and end at NumFields-1.
        For i = 0 To Tbl.NumFields-1 Step 1
            'The FieldNames array holds the names of the fields.
            Print "Field:" & Tbl.FieldNames(i)
            'Print "Formula: " & Tbl.GetFieldFormula(Tbl.FieldNames(i))
            Print "Options: " & Tbl.GetFieldOptions(Tbl.FieldNames(i))
            Print "Size: " & Str(Tbl.GetFieldSize(Tbl.FieldNames(i)))
            Print "Type: " & Str(Tbl.GetFieldType(Tbl.FieldNames(i)))
            Print "===================="
        Next
    End If
End Sub
```

```
' NumJoins property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window.

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' NumPages property
'Returns the number of pages in the active view.
Dim NumPgs As Integer
NumPgs = CurrentApplication.ActiveView.NumPages
```

```
' NumRecordsFound property
'Implement a named find/sort.
CurrentWindow.NamedFindSort="CA"

'Find out how many records were found.
Print CurrentWindow.NumRecordsFound
```

```
' NumRevisions property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window.

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' NumRows method
'This code is pulled from the deleteScheduledRemovedRooms global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim ResSet As New ResultSet

     Dim TName As String
     TName = CurrentDocument.Tables(0).TableName

     If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Qry.Connection = Con
          Qry.TableName = CurrentDocument.Tables(0).Path & TName
               Set ResSet.Query = Qry
               If (ResSet.Execute)Then
                    If (ResSet.NumRows) Then
                         ResSet.FirstRow
                         Do
                              ResSet.DeleteRow
                         Loop While (ResSet.NumRows)
                    End If
               End If
          Con.Disconnect
     End If
End Sub
```

```
' NumTables property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window.

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' NumViews property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window.

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' OnSwitchFromMacro property
'Retrieves the name of the macro to be triggered when the view is
'switched from.
Dim FrmMacro As String
FrmMacro = CurrentApplication.ActiveView.OnSwitchFromMacro

'Or

'Sets the macro to be executed when the view is switched from.
CurrentApplication.ActiveView.OnSwitchFromMacro = "MyMacro"
```

```
' OnSwitchToMacro property
'Retrieves the name of the macro to be triggered when the view is
'switched to.
Dim ToMacro As String
ToMacro = CurrentApplication.ActiveView.OnSwitchToMacro

'Or

'Sets the macro to be executed when the view is switched to.
CurrentApplication.ActiveView.OnSwitchToMacro = "MyMacro"
```

```
' OpenDocument method
'This function lays the groundwork for an application that runs
'in the background. After opening a document, you can hide Approach.

Dim rval As Integer   'Return Value

'Minimize Approach (optional)
CurrentApplication.ApplicationWindow.Minimize

'Hide Approach from the user
CurrentApplication.Visible=False

'Have Approach open an application that runs automatically.
rval = CurrentApplication.OpenDocument("AutoApp", "C:\LOTUS\APPROACH")

'Run your program.

'Show Approach to the user
CurrentApplication.Visible=True

'Maximize Approach (optional)
'CurrentApplication.ApplicationWindow.Maximize

'Quit Approach
CurrentApplication.CloseWindow
```

```
' Orientation property
'Changes the pattern of the line to solid.  The Pattern property accepts
'Approach enumerators for the line styles.
Source.ObjLine.Orientation = $ltsOrientationPosSlope
```

```
' PageBreak property
'This example creates a report, adds a summary panel to the report that is
'grouped by SALESREP and set the summary panel pagebreak property to true so
'that a new page is created for each distinct SALESREP value.

Dim Rpt As Report
Dim Spanel As SummaryPanel
Dim FB As FieldBox
Dim SumFB As FieldBox

Set Rpt = New Report (CurrentDocument, "INVOICE")
Rpt.name="MyReport"
Set Spanel = New SummaryPanel(CurrentDocument.MyReport.body)
Spanel.Height=360
Spanel.background.color.setrgb(COLOR_VANILLA)
Spanel.GROUPBYDATATABLE="INVOICE"
Spanel.GROUPBYDATAFIELD="SALESREP"

Spanel.PAGEBREAK=True

Spanel.name="BySalesRep"
Set FB=New Fieldbox (CurrentDocument.MyReport.BySalesRep)
FB.width=1440
FB.height=360
FB.datatable="INVOICE"
FB.datafield="SALESREP"
CurrentDocument.MyReport.Body.Height=0

Set SumFB= New FieldBox(CurrentDocument.MyReport.BySalesRep)
SumFB.width=1440
SumFB.height=360
SumFB.LEFT=2880
SumFB.datatable="INVOICE"
SumFB.datafield="QtybyRep"
```

```
' Page property
'Retrieves the page number the object is on.
Dim PageNum As Integer
PageNum = Source.Page
```

```
' Parent property
'Print the name of the parent of this object to the output.
Print Source.Parent.Parent.Name
```

```
' Password property
'Sets the password for the document (write-only)
Dim C As New CONNECTION
C.USERID = "UserID"
C.PASSWORD = "UserPassword"
CkConnectTo = C.CONNECTTO("SQL Server",C.USERID ,C.PASSWORD , "sqlsvr_nt351")
C.DISCONNECT
```

```
' Path property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window.

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' Pattern property
'Changes the pattern of the line to solid. The Pattern property accepts
'Approach enumerators for the line styles.
Source.ObjLine.LineStyle.Pattern = $ltsLineStyleSolid
```

```
PrevRecord method
'Go to the previous record.
CurrentWindow.PrevRecord
```

```
' PrevRow Record
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim C As New Connection
     Dim Q As New Query
     Dim RS As New ResultSet

     Dim tname As String
     tname = currentdocument.tables(0).tablename

     If C.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Q.Connection = C
          Q.Tablename = currentdocument.tables(0).path & tname
               Set RS.Query = Q
               If (RS.Execute)Then
                    If (RS.numrows) Then
                         RS.lastrow
                         Do
                              RS.deleteRow

RS.PrevRow
                         Loop While (RS.numrows)
                    End If
               End If
          c.disconnect
     End If
End Sub
```

```
' Query property
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim C As New Connection
     Dim Q As New Query
     Dim RS As New ResultSet

     Dim tname As String
     tname = currentdocument.tables(0).tablename

     If C.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Q.Connection = C
          Q.Tablename = currentdocument.tables(0).path & tname
               Set RS.Query = Q
               If (RS.Execute)Then
                    If (RS.numrows) Then
                         RS.firstrow
                         Do
                              RS.deleteRow
                         Loop While (RS.numrows)
                    End If
               End If
          c.disconnect
     End If
End Sub
```

```
' Quit property
'Quit Approach
CurrentApplication.Quit
```

```
' RadioButtonLabel property
'Retrieve the label for the radio button named ObjRadio.
Dim RBLabel As String
RBLabel  = Source.ObjRadio.LabelText

'Or

'Set the label for the radio button named ObjRadio.
Source.ObjRadio.LabelText = "My Radiobutton Label"
```

```
' ReadOnly property
'Find out if the field named Address is read-only.
If (Source.Address.ReadOnly) Then
   Messagebox("The field is read-only)
Else
   Messagebox("The field is read-write)
Endif

'Or

'Set the Address field to be read-only.
Source.Address.ReadOnly = True
```

```
' Red property
'This is a read-only property. It lets you find out how much red
'is in the color.

Dim b As Long   'Create a variable.

b = source.background.color.red   'Get the amount of red in the background.
Print b   'Print the amount of red.
```

```
' Redraw property
'This property lets you turn redraw off while you move and add objects
'to the form, then redraw the entire form when you are done.

CurrentView.Window.Redraw = False   'Turn off redraw off for the current document.
CurrentView.Body.LastName.Left = 1440   'Move the LastName field.
CurrentView.Body.FirstName.Left = 2880   'Move the FirstName field.
CurrentView.Body.Address.Left = 1440  'Move the Address field.
CurrentView.Body.City.Left = 1440   'Move the City field.
CurrentView.Window.Redraw = True   'Turn redraw back on.
```

```
' Relief property
'Changes the relief of the font to raised. You can use the Approach
'enumerators to designate the type of relief you want.

Source.ObjCheckBox.Relief = $ltsReliefRaised
```

```
' Repaint method
Sub CleanScreen()
     'This program performs the same function
     'that the Clean Screen menu item on the Edit
     'menu does.
     CurrentWindow.Redraw=False   'Turn off redraw temporarily
     'Turn off each bar.
     CurrentWindow.ActionBarVisible=False
     CurrentWindow.IconBarVisible=False
     CurrentWindow.StatusBarVisible=False
     CurrentWindow.ViewTabVisible=False
     CurrentWindow.Redraw=True   'Turn it back on
     CurrentWindow.Repaint   'Now repaint the window.
End Sub
```

```
' Restore method
'Restores the document window.
CurrentApplication.ActiveDocWindow.Restore

'Or

'Restore the Approach Window
CurrentApplication.ApplicationWindow.Restore
```

```
' Right (Border) property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim h As Integer
    Dim i As Integer

    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
End Sub
```

```
' RightMargin property
'Find out what the current right margin setting is for the active view.
Dim RMargin As Long
RMargin = CurrentApplication.ActiveView.RightMargin

'Or

'Set the right margin for the active view.
CurrentApplication.ActiveView.RightMargin = 720
```

```
' RunApproachMacro method
'This code example comes from the click event of the CheckRadio object
'in the Data Entry Screen view in the Checkbook SmartMaster application.

Sub Click(Source As Radiobutton, X As Long, Y As Long)
     source.checknumber.visible=True
     source.depositnumber.visible=False
     RunApproachMacro("SetCheckNumber")
End Sub
```

```
' SendToBack method
'Send the current object to the back (bottom) of the display order.
Source.SendToBack
```

```
' SetCellFocus method
'Set the focus to the current cell in the current worksheet in the column
'you specify by name - Company. For example, you might attach the following
'script to the CellGotFocus event of a worksheet. It will set the focus to
'the current cell in the Company column and then print the text of that
'cell.

Dim Cell As Integer
Cell=CurrentView.SetCellFocus ("Company")
Print Current.View.GetText
```

```
' SetFocus method
'This code example comes from the Enter Date form in the Schedule
'SmartMaster application.

Sub Switchto(Source As Form, View As VIEW)
     source.body.fbxDate.text = ""
     source.body.fbxDate.setfocus
End Sub
```

```
' SetList method
'This global function, modifyRoomsArray, comes from the Schedule
'SmartMaster sample application.

Sub modifyRoomsArray(modifyType As Integer)
     Dim fbx As fieldbox
     Dim lbx As listbox
     Dim btn As button

     Dim i As Integer
     Dim ret As Integer
     Dim roomExists As Integer
     Dim newRoomName As String
     Dim tempRooms() As String
     Dim numRooms As Integer

     If ((Ubound(rooms) + 1) = 20) And (modifyType > 0)Then
          Messagebox "Cannot add anymore rooms.  Maximum number of rooms is 20."
     Else
          Set fbx = currentview.body.fbxRoomName
          Set lbx = currentview.body.lbxRooms
          Set btn = currentview.body.btnDone
          If (modifyType > 0) Then
               roomExists = False
               newRoomName = fbx.text
               For i = 0 To Ubound(rooms)
                    If (Ucase$(rooms(i)) = Ucase$(newRoomName)) Then
                         roomExists = True
                         i = Ubound(rooms) + 1
                    End If
               Next
               If (roomExists = False) Then
                    If (Ubound(rooms) = 0  And rooms(0) = "") Then
                         rooms(0) = newRoomName
                    Else
                         Redim Preserve rooms(Ubound(rooms) + 1)
                         rooms(Ubound(rooms)) = newRoomName
                    End If
                    lbx.setlist rooms
                    fbx.text = ""
                    btn.enabled = True
               Else
                    Messagebox "A room named """ & Ucase$(newRoomName) & """ already
exists"
               End If
          Else
```

```
            ret = Messagebox( "Are you sure you want to delete this room?  Deleting
this room will also delete any scheduled conferences for this room.", 4, "Delete
Room")
            If ret = 6 Then
                deleteRoomName = lbx.text
                Redim Preserve deletedRooms(Ubound(deletedRooms) + 1)
                deletedRooms(Ubound(deletedRooms)) = deleteRoomName
                For i = 0 To Ubound(rooms)
                    If (Ucase$(rooms(i)) = Ucase$(deleteRoomName)) Then
                        numRooms = numRooms - 1
                        For j = i To (Ubound(rooms) - 1)
                            rooms(j) = rooms(j + 1)
                        Next
                        i = Ubound(rooms) + 1
                    End If
                Next
                If (Ubound(rooms)) Then
                    Redim Preserve rooms(Ubound(rooms) - 1)
                Else
                    rooms(0) = ""
                    currentview.body.btnRemove.enabled = False
                End If
            End If
        End If
        lbx.setlist rooms
        btn.enabled = True
        string_sort rooms
    End If

End Sub
```

```
' SetPicture method
'Set the picture in the picture object to the TILES.BMP in the Win95 'directory.
Source.ObjPicture.SetPicture("c:\win95\tiles.bmp")
```

```
' SetRGB
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
     Dim tt As textbox

     Dim t As Integer
     Dim i As Integer

     For i = 0 To Ubound(rooms)
          If rooms(i) = roomName Then
               t = i
               i = Ubound(rooms)
          End If
     Next
     t = 1635 + (330 * t)
     Set tt = New textbox(currentview.body)
     tt.text = " " & txt & " "
     tt.font.size = 8
     tt.border.style = $ltsBorderStyleNone
     tt.border.left = True
     tt.border.right = True
     tt.border.top = False
     tt.border.bottom = False
     tt.border.width = $apr1point
     tt.border.color.setrgb color_ultramarine
     tt.background.color.setrgb color_50_gray
     tt.height = 325
     tt.top = t
     tt.left = (((start - 8) * 750) + 960)
     tt.width =  (750 * (finish - start))
     tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' SetValue
'This is the modifyRooms global function from the Schedule
'SmartMaster application.

Function modifyRooms
     Dim C As New Connection
     Dim Q As New Query
     Dim RS As New ResultSet

     modifyRooms = False

     If C.ConnectTo("dBASE IV") Then
          Set Q.Connection = C
          Q.Tablename = currentdocument.tables(0).path & "rooms"
          Set RS.Query = Q
          If (RS.Execute)Then
               If (RS.numrows) Then
                    RS.firstrow
                    Do
                         RS.deleteRow
                    Loop While (RS.numrows)
               End If
               modifyRooms = True
               For i = 0 To Ubound(rooms)
                    RS.addrow
                    RS.setvalue "room", rooms(i)
                    RS.updaterow
               Next
          End If
     End If
     c.disconnect

End Function
```

```
' ShadowColor property
'Set the shadow color of the current object to the color red.
'Create a color object named Red.
Dim Red As New color(255,0,0)
Set Source.ShadowColor = Red
```

```
' ShowArrow property
'Displays an arrow on the dropdownbox.
Sub Click(Source As Textbox, X As Long, Y As Long)
    Source.CustName.ShowArrow = True
End Sub
```

```
' ShowAsDialog property
'Finds out if the form is displayed as a dialog box.
Dim rval As Integer
rval = CurrentApplication.ActiveView.ShowAsDialog

'Or

'Specify that the current form is displayed as a dialog box.
CurrentApplication.ActiveView.ShowAsDialog = True
```

```
' ShowInPreview property
'Set this object not to show in Preview mode.
Source.ShowInPreview = False

'Or

'Find out if this object is set to show in Preview.
Dim ShowInPrev As Integer
ShowInPrev = Source.ShowInPreview
```

```
' Size property
Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim t As Integer
    Dim i As Integer

    For i = 0 To Ubound(rooms)
        If rooms(i) = roomName Then
            t = i
            i = Ubound(rooms)
        End If
    Next
    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
    tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' SlideLeft property
'Set this object to slide to the left when printing (if space permits).
Source.SlideLeft = True

'Or

'Find out if this object is set to slide left when printing.
Dim rval As Integer
rval = Source.SlideLeft
```

```
' SlideUp property
'Set this object to slide up when printing if space permits.
Source.SlideUp = True

'Or

'Find out if this object is set to slide up when printing.
Dim rval As Integer
rval = Source.SlideUp
```

```
' SQL property
'This code is pulled from the deleteScheduledRemovedRooms global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim ResSet As New ResultSet

     Dim TName As String
     TName = CurrentDocument.Tables(0).TableName

     If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Qry.Connection = Con
          Qry.TableName = CurrentDocument.Tables(0).Path & TName
          For i = 1 To Ubound(DeletedRooms)
               Qry.SQL = "SELECT * FROM """ & Qry.TableName & """" & TName & " WHERE ("&
TName & "."Room Name/Number"" = '" & DeletedRooms(i) & "')"
               Set ResSet.Query = Qry
               If (ResSet.Execute)Then
                    If (ResSet.NumRows) Then
                         ResSet.FirstRow
                         Do
                              ResSet.DeleteRow
                         Loop While (ResSet.NumRows)
                    End If
               End If
          Next
          Con.Disconnect
     End If
End Sub
```

```
' StatusBarVisible property
Sub CleanScreen()
     'This program performs the same function
     'that the Clean Screen menu item on the Edit
     'menu does.
     CurrentWindow.Redraw=False   'Turn off redraw temporarily
     'Turn off each bar.
     CurrentWindow.ActionBarVisible=False
     CurrentWindow.IconBarVisible=False
     CurrentWindow.StatusBarVisible=False
     CurrentWindow.ViewTabVisible=False
     CurrentWindow.Redraw=True   'Turn it back on
     CurrentWindow.Repaint   'Now repaint the window.
End Sub
```

```
' Strikethru property
'Find out if strikethrough is displayed.
Dim rval As Integer
rval = Source.Font.StrikeThrough

'Or

'Set the strikethrough to display.
Source.Font.StrikeThrough = True
```

```
' Style property
Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim t As Integer
    Dim i As Integer

    For i = 0 To Ubound(rooms)
        If rooms(i) = roomName Then
            t = i
            i = Ubound(rooms)
        End If
    Next
    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
    tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' TableName property
'This code is pulled from the 'deleteScheduledRemovedRooms' global
'function in the Schedule application.

Sub deleteScheduledRemovedRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim ResSet As New ResultSet

     Dim TName As String
     TName = CurrentDocument.Tables(0).TableName

     If Con.ConnectTo("dBASE IV") Then   'Connect to dBASE.
          Set Qry.Connection = Con
          Qry.TableName = CurrentDocument.Tables(0).Path & TName
          For i = 1 To Ubound(DeletedRooms)
               Qry.SQL = "SELECT * FROM """ & Qry.TableName & """" & TName & " WHERE ("&
TName & "."Room Name/Number"" = '" & DeletedRooms(i) & "')"
               Set ResSet.Query = Qry
               If (ResSet.Execute)Then
                    If (ResSet.NumRows) Then
                         ResSet.FirstRow
                         Do
                              ResSet.DeleteRow
                         Loop While (ResSet.NumRows)
                    End If
               End If
          Next
          Con.Disconnect
     End If
End Sub
```

```
' Tables property
'We need to access the main table through the data object.
'The first step is to find out the name of the main table.
Dim MnTbl As String
Dim NumTbls As Integer
Dim Tbls As Variant
Dim t As Variant

MnTbl = CurrentView.MainTable   'Get the name of the main table.
Set Tbls = CurrentDocument.Tables   'Get the collection of tables.
NumTbls = Tbls.Count   'Find out how many tables there are.
For i = 1 To NumTbls Step 1
    If (Tbls(i-1).TableName = MnTbl) Then   'If we've found the right one.
        'Data access code goes here
        j=1
    End If
Next
```

```
' TabNext
'Sets focus to the next field in the Tab Order.
CurrentWindow.TabNext
```

```
' TabOrder property
'Print the tab order of the object to the output.
Print Source.TabOrder

'Or

'Set the tab order for this object.
Source.TabOrder = 5
```

```
' TabPrev method
'Sets focus to the previous field in the Tab Order.
CurrentWindow.TabPrev
```

```
' TabStop property
'Sets whether users can tab to this object or not.
'Users can tab to this object.
Source.TabStop = True

'Or

'Gets if this object can be tabbed to.
Dim rval As Integer
rval = Source.TabStop
```

```
' Text property
Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim t As Integer
    Dim i As Integer

    For i = 0 To Ubound(rooms)
        If rooms(i) = roomName Then
            t = i
            i = Ubound(rooms)
        End If
    Next
    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
    tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' Text (Button) property
'Find out what text displays on the button named ObjButton.
Dim Txt As String
Txt = Source.ObjButton.Text

'Or

'Set the text to display on the button named ObjButton.
Source.ObjButton.Text = "Push Me"
```

```
' Tile method
'Tile the open windows.
CurrentApplication.ApplicationWindow.Tile
```

```
TimerInterval
'Find out what the timer interval for the current view is set to.
Dim Tmr As Long
Tmr = CurrentApplication.ActiveView.TimerInterval

'Or

'Set the timer interval for the current view.
CurrentApplication.ActiveView.TimerInterval = 300
```

```
' Top property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
    Dim tt As textbox

    Dim t As Integer
    Dim i As Integer

    For i = 0 To Ubound(rooms)
        If rooms(i) = roomName Then
            t = i
            i = Ubound(rooms)
        End If
    Next
    t = 1635 + (330 * t)
    Set tt = New textbox(currentview.body)
    tt.text = " " & txt & " "
    tt.font.size = 8
    tt.border.style = $ltsBorderStyleNone
    tt.border.left = True
    tt.border.right = True
    tt.border.top = False
    tt.border.bottom = False
    tt.border.width = $apr1point
    tt.border.color.setrgb color_ultramarine
    tt.background.color.setrgb color_50_gray
    tt.height = 325
    tt.top = t
    tt.left = (((start - 8) * 750) + 960)
    tt.width =  (750 * (finish - start))
    tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' Top (Border)property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
     Dim tt As textbox

     Dim t As Integer
     Dim i As Integer

     For i = 0 To Ubound(rooms)
          If rooms(i) = roomName Then
               t = i
               i = Ubound(rooms)
          End If
     Next
     t = 1635 + (330 * t)
     Set tt = New textbox(currentview.body)
     tt.text = " " & txt & " "
     tt.font.size = 8
     tt.border.style = $ltsBorderStyleNone
     tt.border.left = True
     tt.border.right = True
     tt.border.top = False
     tt.border.bottom = False
     tt.border.width = $apr1point
     tt.border.color.setrgb color_ultramarine
     tt.background.color.setrgb color_50_gray
     tt.height = 325
     tt.top = t
     tt.left = (((start - 8) * 750) + 960)
     tt.width =  (750 * (finish - start))
     tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' Type property
'In this code example we're going to create a listbox and fill
'the listbox with the views available.
Dim LstBox As ListBox
Dim aryVws(20) As String
Dim NumVws As Integer

'Check to find out if the current view is a form.
If (CurrentView.Type = $aprForm) Then
    'Create a new listbox on the current view
    Set LstBox = New ListBox(CurrentView.Body, 1)
    NumVws = CurrentDocument.Views.Count   'Get the number of views
    For i = 0 To NumVws-1   'Fill the array with the view names.
        aryVws(i) = CurrentDocument.Views(i).Name
    Next
    LstBox.SetList(aryVws)   'Set the listbox with the list of views.
Else
    Messagebox "You must be on a form."
End If
```

```
' User property
Sub DocumentReport()
    'This function prints a report of all of the document information
    'to the output window.

    'Print each of the items to the output.
    Print "Author: " & CurrentDocument.Author
    Print "Description: " & CurrentDocument.Description
    Print "Keywords: " & CurrentDocument.Keywords
    Print "User: " & CurrentDocument.User

    Print "FileName: " & CurrentDocument.Filename
    Print "FullName: " & CurrentDocument.FullName
    Print "Path of the .APR: " & CurrentDocument.Path

    Print "Creation Date: " & CurrentDocument.CreateDate
    Print "LastModified: " & CurrentDocument.LastModified

    If (CurrentDocument.Modified) Then   'If the document has been modified...
        Print "The document has been modified: " & Str(CurrentDocument.NumRevisions)
& " times."
    Else
        Print "The document hasn't been modified."
    End If
    Print "Number of joins: " & Str(CurrentDocument.NumJoins)
    Print "Number of tables in the .APR: " & Str(CurrentDocument.NumTables)
    Print "Number of views in the .APR: " & Str(CurrentDocument.NumViews)
End Sub
```

```
' Underline property
'Find out if the text in the LastName field is underlined.
Dim Underln As Integer
Underln = Source.LastName.Font.Underline

'Or

'Set the text in the LastName field to be underlined.
Source.LastName.Font.Underline = True
```

```
' UpdateRow
'This is the modifyRooms global function from the Schedule
'SmartMaster application.

Function modifyRooms
     Dim Con As New Connection
     Dim Qry As New Query
     Dim ReSet As New ResultSet

     modifyRooms = False

     If Con.ConnectTo("dBASE IV") Then
          Set Qry.Connection = Con
          Qry.Tablename = currentdocument.tables(0).path & "rooms"
          Set ReSet.Query = Qry
          If (ReSet.Execute)Then
               If (ReSet.numrows) Then
                    ReSet.firstrow
                    Do
                         ReSet.deleteRow
                    Loop While (ReSet.numrows)
               End If
               modifyRooms = True
               For i = 0 To Ubound(rooms)
                    ReSet.addrow
                    ReSet.setvalue "room", rooms(i)
                    ReSet.updaterow
               Next
          End If
     End If
     con.disconnect

End Function
```

```
' VarTable property
'Retrieve the VarTable for the current document.
Dim Tbl As Table
Set Tbl = CurrentApplication.ActiveDocument.VarTable
```

```
' Views property
'In this code example we're going to create a listbox and fill
'the listbox with the views available.
Dim LstBox As ListBox
Dim aryVws(20) As String
Dim NumVws As Integer

'Check to find out if the current view is a form.
If (CurrentView.Type = $aprForm) Then
     'Create a new listbox on the current view
     Set LstBox = New ListBox(CurrentView.Body, 1)
     NumVws = CurrentDocument.Views.Count   'Get the number of views
     For i = 0 To NumVws-1   'Fill the array with the view names.
          aryVws(i) = CurrentDocument.Views(i).Name
     Next
     LstBox.SetList(aryVws)   'Set the listbox with the list of views.
Else
     Messagebox "You must be on a form."
End If
```

```
' ViewTabVisible property
Sub CleanScreen()
    'This little program performs the same function
    'that the Clean Screen menu item on the Edit
    'menu does.
    CurrentWindow.Redraw=False   'Turn off redraw temporarily
    'Turn off each bar.
    CurrentWindow.ActionBarVisible=False
    CurrentWindow.IconBarVisible=False
    CurrentWindow.StatusBarVisible=False
    CurrentWindow.ViewTabVisible=False
    CurrentWindow.Redraw=True   'Turn it back on
    CurrentWindow.Repaint   'Now repaint the window.
End Sub
```

```
' Visible property
'This global function comes from the Checkbook SmartMaster application.

Sub SetDefaultButtonText
     If currentview.body.commitflag.text="1" Then
          currentview.body.voidbutton.text="Remove this transaction from the balance"
     Else
          currentview.body.voidbutton.text="Apply this transaction to the balance"
     End If
     If currentview.body.Transtype.text="Check" Then
          currentview.body.Checknumber.visible=True
          currentview.body.Depositnumber.visible=False
     End If
     If currentview.body.Transtype.text="Deposit" Then
          currentview.body.Checknumber.visible=False
          currentview.body.Depositnumber.visible=True
     Else
          currentview.body.Checknumber.visible=False
          currentview.body.Depositnumber.visible=False
     End If
End Sub
```

```
' Width property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
     Dim tt As textbox

     Dim t As Integer
     Dim i As Integer

     For i = 0 To Ubound(rooms)
          If rooms(i) = roomName Then
               t = i
               i = Ubound(rooms)
          End If
     Next
     t = 1635 + (330 * t)
     Set tt = New textbox(currentview.body)
     tt.text = " " & txt & " "
     tt.font.size = 8
     tt.border.style = $ltsBorderStyleNone
     tt.border.left = True
     tt.border.right = True
     tt.border.top = False
     tt.border.bottom = False
     tt.border.width = $apr1point
     tt.border.color.setrgb color_ultramarine
     tt.background.color.setrgb color_50_gray
     tt.height = 325
     tt.top = t
     tt.left = (((start - 8) * 750) + 960)
     tt.width =  (750 * (finish - start))
     tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' Width(Border) property
'This example comes from the displayBlock global function
'in the Schedule SmartMaster application.

Sub displayBlock(txt As String, start As Double, finish As Double, roomName As String)
     Dim tt As textbox

     Dim t As Integer
     Dim i As Integer

     For i = 0 To Ubound(rooms)
          If rooms(i) = roomName Then
               t = i
               i = Ubound(rooms)
          End If
     Next
     t = 1635 + (330 * t)
     Set tt = New textbox(currentview.body)
     tt.text = " " & txt & " "
     tt.font.size = 8
     tt.border.style = $ltsBorderStyleNone
     tt.border.left = True
     tt.border.right = True
     tt.border.top = False
     tt.border.bottom = False
     tt.border.width = $apr1point
     tt.border.color.setrgb color_ultramarine
     tt.background.color.setrgb color_50_gray
     tt.height = 325
     tt.top = t
     tt.left = (((start - 8) * 750) + 960)
     tt.width =  (750 * (finish - start))
     tt.name = "tt" & Str$(tt.top) & Str$(tt.left)
End Sub
```

```
' Window property
'This property lets you turn redraw off while you move and add objects
'to the form, then redraw the entire form when your done.

CurrentView.Window.Redraw = False   'Turn off redraw off for the current document.
CurrentView.Body.LastName.Left = 1440   'Move the LastName field.
CurrentView.Body.FirstName.Left = 2880   'Move the FirstName field.
CurrentView.Body.Address.Left = 1440   'Move the Address field.
CurrentView.Body.City.Left = 1440  'Move the City field.
CurrentView.Window.Redraw = True   'Turn redraw back on.
```

```
' Windows property
'Retrieve the collection of Windows on the Application.
Dim Wins As Variant
Set Wins = CurrentApplication.Windows
```

```
' Yellow property
'This is a read-only property.  It lets you find out how much yellow
'is in the color.

Dim b As Long   'Create a variable.

b = source.background.color.yellow   'Put the amount of yellow in the background.
Print b   'Print the amount of yellow.
```

Approach: CalcTable property

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

(read-only) Returns information for all calculated fields defined for the Approach file. There is only one CalcTable per document.

Data type

Table

Syntax

tablename = *documentname*.**CalcTable**

Legal values

The legal value for the CalcTable property is CalTable. The default value for the CalcTable property is the calculation table for the current document.

Approach: CreateDate property

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

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

(read-only) Returns the date and time the document was created.

Data type

Variant

Syntax

date/time = *documentname*.**CreateDate**

Legal values

The default values are the date and time the document was created.

Approach: CurrentRecord property

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

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

Sets and returns the record number of the current record.

Data type

Long

Syntax

docwindowname.**CurrentRecord** = *recordnumber*

recordnumber = *docwindowname*.**CurrentRecord**

Legal values

You can set this property to any integer between 1 to the total number of records in the database.

Approach: Description property

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

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

Sets or returns the description of the document.

Data type

String

Syntax

documentname.**Description** = *stringexp*

stringexp = *documentname*.**Description**

Legal values

Any string up to 256? characters. The default value for the Description property is Blank.

Approach: Dispatch property

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

(read-only) Returns the methods and properties of an OLE object if it supports OLE Automation. For example, you might have a word processer document embedded in an Approach form, and the word processor server supports OLE Automation.

Data type

Variant

Syntax

value = *oleobjectname*.**Dispatch**

Legal values

TBD

Approach: Display property

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

Sets or returns the display behavior of the data stored in PicturePlus fields.

Data type

Long

Syntax

ppobject.**Display** = *value*

value = *documentname*.**Description**

Legal values

Value	Description
$ItsPictureDisplayCrop	(Default) Crop the picture to the dimension of the field box.
$ItsPictureDisplayShrink	Shrink the picture to fit in the field box

Approach: FieldNames property

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

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

(read-only) Returns a list (string arrary) of the fields defined in the current database (table).

Data type

Variant

Syntax

*dbname**.FieldNames** = list*

Legal values

Any field in the database.

Approach: FileName property

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

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

(read-only) Returns the name of the current database, including path and extension.

Data type

String

Syntax

stringexp = *dbname*.**Filename**

Legal values

The drive, directory, filename, and extension of the current database.

Approach: FullName property

{button ,AL(`H_APPLICATION_CLASS;H_DOCUMENT_CLASS;H_TABLE_CLASS;',0)} See list of classes

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

(read-only) Returns the path of the Approach executable, the APR file, or the database file.

Data type

String

Syntax

stringexp = *appname*.**Fullname**

or

stringexp = *docname*.**Fullname**

or

stringexp = *tablename*.**Fullname**

Legal values

Value	Description
drive:\directory\ filename.exe	(Default) The drive and directory of the Approach executable file.
drive:\directory\ filename.apr	The drive and directory of the Approach document file.
drive:\directory\ dbname.*	The drive and directory of the database file.

Approach: GroupByDataField property

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

Sets or returns the field used in a summary calculation across the group of records.

Data type

String

Syntax

summarypanel.**GroupByDataField** = *stringexp*

stringexp = *summarypanel*.**GroupByDataField**

Legal values

The legal values are any field in the database.

Approach: GroupByDataTable property

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

Sets or returns the table containing the field used in a summary calculation across the group of records specified in GroupBy.

Data type

String

Syntax

summarypanel.**GroupByDataTable** = *stringexp*

stringexp = *summarypanel*.**GroupByDataTable**

Legal values

The legal values are any table in the document.

Approach: GroupByEvery property

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

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

Sets or returns the number of records you want to group in summary calculations.

Data type

Integer

Syntax

summarypanel.**GroupByEvery** = *value*

value = *summarypanel*.**GroupByEvery**

Legal values

You can set this property up to the total number of records in the database.

Approach: IconBarVisible property

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

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

Sets or returns whether the icon bar is visible.

Data type

Integer

Syntax

docwindowname.**IconBarVisible** = *flag*

flag = *docwindowname*.**IconBarVisible**

Legal values

Value	Description
FALSE	The icon bar is not visible.
TRUE	The icon bar is visible.

Approach: IconSets property

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

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

(read-only) Returns the iconsets available for an application.

Data type

Variant

Syntax

stringarray = *applicationobjectname*.**IconSets**

Legal values

The names of the Icon bars in the application.

Approach: IsClicked property

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

Sets or returns whether the radiobutton is switched on.

Data type

Integer

Syntax

flag = *radobuttonname*.**IsClicked**

radobuttonname.**IsClicked** = *flag*

Legal values

Value	Description
TRUE	The radiobutton is switched on.
FALSE	The radiobutton is switched off.

Approach: IsConnection property

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

(Read-only) Returns whether the connection to the backend is still active. For example Lotus Notes, Oracle.

Data type

Integer

Syntax

flag = *dbname*.**IsConnection**

Legal values

Value	Description
TRUE	This is is a connection type table.
FALSE	This is a local type table.

Approach: KeepRecsTogether property

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

Data type

Integer

Syntax

flag = *reportname*.**KeepRecsTogether**

reportname.**KeepRecsTogether** = *flag*

Approach: Keywords property

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

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

Sets or returns the list of keywords in Approach File Properties, used to identify the document.

Data type

String

Syntax

docname.**Keywords** = *stringexp*

stringexp = *docname*.**Keywords**

Legal values

You can set this property to any string up to ? characters.

Approach: Language property

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

(read-only) Returns the language used in the current application as a 2 character code, for example EN for English, FR for French.

Data type

String

Syntax

stringexp = *applicationname*.**Language**

Legal values

All the 2 character language codes.

Approach: LastModified property

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

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

(read-only) Returns the date and time the document was last modified in seconds.

Data type

Variant

Syntax

variant = *docname*.**LastModified**

Legal values

The legal value for the LastModified property is the date and time the document was last modified.

Approach: Location property

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

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

Sets or returns whether the summary is a leading or trailing summary panel.

Data type

Long

Syntax

panelname.**Location** = *value*

value = *panelname*.**Location**

Legal values

Value	Description
aprSummaryTrailing	The summary is a leading summary panel.
aprSummaryLeading	The summary is a trailing summary panel

Approach: LPObject property

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

Sets or returns the value of the C pointer to the OleObject interface of the OLE object, or 0 if there is no connection to the OLE object yet. You can pass this value to support DLL's for advanced OLE interface access. The lock on the OLE object is not incremented when this property is accessed (i.e., AddRef has not been called).

Data type

Long

Syntax

value = *oleobjectame*.**LPObject**

oleobjectame.**LPObject** = *value*

Approach: MainTable property

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABEL_CLASS;H_REPEATINGPANEL_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

Sets or returns the main table for the view.

Data type

String

Syntax

viewname.**MainTable** = *stringexp*

stringexp = *viewname*.**MainTable**

Legal values

You can set the property to any database in the document.

Approach: MenuBar property

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABEL_CLASS;H_REPEATINGPANEL_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

Sets or returns the menu items in the current view.

Data type

String

Syntax

viewname.**MenuBar** = *stringexp*

stringexp = *viewname*.**MenuBar**

Approach: Menus property

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

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

(read-only) Returns an array of menu names.

Data type

Variant

Syntax

stringexp = *appname*.**Menus**

or

stringexp = *docname*.**Menus**

Legal values

For the application, the default set of menus (Default Menu + Short Menu). For the document, all the names of the user-defined custom menus.

Approach: Modified property

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

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

(read-only) Returns whether the document has been modified.

Data type

Integer

Syntax

flag = *docname*.**Modified**

Legal values

Value	Description
TRUE	The document has been modified.
FALSE	The document has not been modified.

Approach: NamedFindSort property

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

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

Sets or returns the name of the current Named Find/Sort.

Data type

String

Syntax

docwindowobjectname.**NamedFindSort** = *stringexp*

stringexp = *docwindowobjectname*.**NamedFindSort**

Legal values

You can set the property any of the Named Find/Sorts in the document.

Approach: NamedFindSorts property

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

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

(read-only) Returns the list of Named Find/Sorts in the document.

Data type

Variant

Syntax

stringarray = *docname*.**NamedFindSorts**

Approach: NamedStyles property

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

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

(read-only) Returns a list of the named styles in the document.

Data type

Variant

Syntax

stringarray = *docname*.**NamedStyles**

Legal values

All the named styles in the document.

Approach: NumColumns property

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

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

Sets or returns the number of columns displayed in the current report.

Data type

Integer

Syntax

reportname.**NumColumns** = *value*

value = *reportname*.**NumColumns**

Approach: NumFields property

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

(read-only) Returns the number of fields in the table.

Data type

Integer

Syntax

value = *dbname*.**NumFields**

Approach: NumJoins property

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

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

(read-only) Returns the number of joins in the document.

Data type

Integer

Syntax

value = *docname*.**NumJoins**

Approach: NumLines property

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

Sets or returns the number of lines in a repeating panel.

Data type

Integer

Syntax

repeatingpanelname.**NumLines** = *value*

value =*repeatingpanelname*.**NumLines**

Legal values

You can set this property to any integer between 1 and 30.

Approach: NumPages property

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

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

(read-only) Returns the number of pages of a multipage form or form letter.

Data type

Integer

Syntax

value = *formname*.**NumPages**

or

value = *formlettername*.**NumPages**

Legal values

The default value fo rthe NumPages property is the number of pages in the form.

Approach: NumRecords property

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

(read-only) Returns the number of records in the table.

Data type

Long

Syntax

value = *tablename*.**NumRecords**

Legal values

The default value fo rthe NumRecords property is the number of records in the table.

Approach: NumRecordsFound property

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

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

(read-only) Returns the number of records in the found set.

Data type

Long

Syntax

value = *foundset*.**NumRecordsFound**

Legal values

The default value fo rthe NumRecordsFound property is the number of records in the found set.

Approach: NumRevisions property

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

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

(read-only) Returns the number of times the document has been updated.

Data type

Long

Syntax

value = *docname*.**NumRevisions**

Legal values

The default value for the NumRevisions property is the number of times it has been revised.

Approach: NumTables property

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

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

(read-only) Returns the number of open tables in the document.

Data type

Integer

Syntax

value = *docname*.**NumTables**

Legal values

The default value for the NumTables property is the number of tables in the document.

Approach: NumViews property

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

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

(read-only) Returns the number of views in the document.

Data type

Integer

Syntax

value = *docname*.**NumViews**

Legal values

The default value for the NumViews property is the number of views in the document.

Approach: ObjectList property

{button ,AL(`H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABEL_CLASS;H_REPORT_CLASS;',0)} See list of classes

(read-only) Returns a collection of all the display objects in the view.

Data type

Collection

Syntax

set collection = *viewname*.**ObjectList**

Legal values

The default value for the ObjectList property is the list of display objects in the document.

Approach: OnSwitchFromMacro property

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABELS_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

Sets or returns the name of the macro you want to execute when you switch from this view.

Data type

String

Syntax

viewname.**OnSwitchFromMacro =** *stringexp*

stringexp = *viewname*.**OnSwitchFromMacro**

Legal values

You can set this property to any Approach macro in the current document.

Approach: OnSwitchToMacro property

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABELS_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

Sets or returns the name of the macro you want to execute when you switch to this view.

Data type

String

Syntax

viewname.**OnSwitchToMacro =** *stringexp*

stringexp = *viewname*.**OnSwitchToMacro**

Legal values

You can set this property to any Approach macro in the current document.

Approach: PageBreak property

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

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

Sets or returns a page break.

Data type

Integer

Syntax

summarypanelname.**PageBreak =** *flag*

flag = summarypanelname.**PageBreak**

Legal values

Value	Description
TRUE	Insert a page break
FALSE	Do not insert a page break

Approach: Password property

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

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

(write-only) Sets a password used for connecting to a server. If there is no password, the Login dialog appears.

Data type

String

Syntax

connection.**Password** = *stringexp*

Legal values

You can set this property to any string up to ? characters.

Approach: Path property

{button ,AL(`H_APPLICATION_CLASS;H_DOCUMENT_CLASS;H_TABLE_CLASS;',0)} See list of classes

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

(read-only) Returns the drive and directory of the application executable file (APPROACH.EXE), current document, or table.

Data type

String

Syntax

stringexp = *appname*.**Path**

or

stringexp = *docname*.**Path**

or

stringexp = *dbname*.**Path**

Legal values

Any drive or directory that conforms to DOS naming conventions.

Approach: PrintDate property

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

Sets or returns whether the date is printed on a worksheet or crosstab printout.

Data type

Integer

Syntax

viewname.**PrintDate =** *flag*

flag = *viewname*.**PrintDate**

Legal values

Value	Description
TRUE	Print the date.
FALSE	Do not print the date.

Approach: PrintPageNum property

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

Sets or returns whether the page number is printed on a worksheet or crosstab printout.

Data type

Integer

Syntax

viewname.**PrintPageNum =** *flag*

flag = *viewname*.**PrintPageNum**

Legal values

Value	Description
TRUE	Print the page number.
FALSE	Do not print the page number.

Approach: PrintTitle property

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

Sets or returns whether the title is printed on a worksheet or crosstab printout.

Data type

Integer

Syntax

viewname.**PrintTItle =** *flag*

flag = *viewname*.**PrintTItle**

Legal values

Value	Description
TRUE	Print the title.
FALSE	Do not print the title.

Approach: Redraw property

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

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

Sets or returns whether the screen is redrawn each time something changes. If you turn this off, you can add and remove many objects at the same time, then repaint the screen, to reduce flickering.

Data type

Integer

Syntax

docwindowname.**Redraw =** *flag*

flag = *docwindowname*.**Redraw**

Legal values

Value	Description
TRUE	Redraw the screen.
FALSE	Do not redraw the screen.

Approach: Selection property

{button ,AL(`H_APPLICATION_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABELS_CLASS;',0)} See list of classes

Sets or returns the current selection object, or null. For example, in Browse if you have focus on a fieldbox, it returns that object.

Data type

Variant

Syntax

Selection = *objectname*

objectname = **Selection**

Approach: ShowRelated property

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

Sets or returns whether related rows and columns appear in crosstabs.

Data type

Integer

Syntax

viewname.**ShowRelated =** *flag*

flag = *viewname*.**ShowRelated**

Legal values

Value	Description
TRUE	Show related rows and columns in crosstabs.
FALSE	Do not show related rows and columns in crosstabs.

Approach: StatusBarVisible property

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

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

Sets or returns whether the status bar is hidden.

Data type

Integer

Syntax

docwindowname.**StatusBarVisible =** *flag*

flag = *docwindowname*.**StatusBarVisible**

Legal values

Value	Description
TRUE	The status bar is visible.
FALSE	The status bar is hidden.

Approach: TableName property

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

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

Sets or returns the drive, directory, filename and extension of the local table, or the servername, directory, filename and extension of the data sources to which Approach can connect.

Data type

String

Syntax

queryobjectname.**TableName =** *stringexp*

stringexp = *queryobjectname*.**TableName**

Legal values

Any table name, including the path.

Approach: Tables property

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

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

(read-only) Returns the tables open in the document.

Data type

BaseCollection

Syntax

set basecollection = *docname*.**Tables**

Legal values

All the tables in the document.

Approach: TimerInterval property

{button ,AL(`H_CHARTVIEW_CLASS;H_CROSSTAB_CLASS;H_ENVELOPE_CLASS;H_FORMLETTER_CLASS;H_FORM_CLASS;H_MAILINGLABEL_CLASS;H_REPORT_CLASS;H_VIEW_CLASS;H_WORKSHEET_CLASS;',0)} See list of classes

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

Sets or returns the Usertimer event in milliseconds.

Data type

Long

Syntax

viewname.**TimerInterval =** *flag*

flag = *viewname*.**TimerInterval**

Legal values

You can set this property to any number greater than or equal to 200. To turn off the timer, set the property to zero.

Approach: Title property

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

Sets or returns the title of the worksheet or crosstab.

Data type

String

Syntax

*viewname***.Title =** *stringexp*

stringexp = *viewname***.Title**

Legal values

You can set this property to any string up to ? characters.

Approach: Transparent property

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

(read-only) Returns whether the color is transparent.

Data type

Integer

Syntax

value = *colorname*.**Transparent**

Legal values

255 means the color is transparent.

Approach: User property

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

(read-only) Returns the current user or group name who is logged into the document.

Data type

String

Syntax

stringexp = *documentname*.**User**

Approach: VarTable property

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

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

(read-only) Returns the variable table for the document. The table contains all the variable fields for the document.

Data type

Table

Syntax

set table = *docame*.**VarTable**

Approach: Views property

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

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

(read-only) Returns the collection of all the views in the document.

Data type

BaseCollection

Syntax

basecollectionname = *docname*.**Views**

Approach: ViewTabVisible property

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

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

Sets or returns whether the view tabs are visible.

Data type

Integer

Syntax

docwindowname.**ViewTabVisible =** *flag*

flag = *docwindowname*.**ViewTabVisible**

Legal values

Value	Description
TRUE	(Default) Display the view tabs.
FALSE	Do not display the view tabs.

Approach: Window property

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

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

(read-only) Returns the document window for this document.

Data type

DocWindow

Syntax

object = *docobject*.**Window**

Approach: Windows property

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

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

(read-only) Returns a collection of all DocWindows controlled by Approach.

Data type

BaseCollection

Syntax

basecollection = *applicationname*.**Windows**

Legal values

The default value for the Windows property is all windows controlled by Approach.

Approach LotusScript A-Z

A

B

C

D

J

(None)

K

L

M

N

T

U

V

W

X

(None)

Y

Z

(None)

Approach LotusScript Classes A-Z

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

A

Application class
ApplicationWindow class

B

Background class
BaseCollection class
BodyPanel class
Border class
Button class

C

ChartView class
CheckBox class
Collection class
Color class
Connection class
Crosstab class

D

Display class
Document class
DocWindow class
DropdownBox class

E

Ellipse class
Envelope class

F

FieldBox class

Font class
Form class
FormLetter class

G

(None)

H

HeaderFooterPanel class

I

(None)

J

(None)

K

(None)

L

LineObject class
LineStyle class
ListBox class

M

MailingLabels class

N

(None)

O

(None)

P

Panel class
Picture class
PicturePlus class

Q

Query class

R

RadioButton class
Rectangle class
RepeatingPanel class
Report class
ResultSet class
RoundRect class

S

SummaryPanel class

T

Table class
TextBox class

U

(None)

V

View class

W

Window class

X

(None)

Y

(None)

Z

(None)

Approach LotusScript Properties A-Z

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

A

B

C

M

N

O

P

X

(None)

Y

Z

(None)

Approach LotusScript Methods A-Z

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

A

Activate method
Add method (Collection class)
AddColumn method
AddRow method

B

BringToFront method

C

Cascade method
Close method
Close method (ResultSet class)
CloseWindow method
ConnectTo method
CreateResultSet method

D

DeletePage method
DeleteRow method
Disconnect method
DoMenuCommand method

E

Execute method

F

FieldExpectedDataType method
FieldID method
FieldName method
FieldNativeDataType method
FieldSize method

Approach LotusScript Events A-Z

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

A

(None)

B

Broadcast event

C

CellDataChange event
CellGetFocus event
CellLostFocus event
Change event
Click event
Closewindow event

D

DocumentClose event
DocumentCreated event
DocumentOpened event
DoubleClick event

E

(None)

F

(None)

G

GotFocus event

H

(None)

I

(None)

J

(None)

K

KeyDown event
KeyPress event
KeyUp event

L

LostFocus event

M

MailCheck event
MailReceived event
MailSend event
MouseDown event
MouseOver event
MouseUp event

N

NewRecord event

O

OpenWindow event

P

(None)

Q

Quit event

R

RecordChange event

S

SelectColumn event
SwitchFrom event
SwitchTo event

T

(None)

U

UserTimer event

V

ViewSwitch event

W

(None)

X

(None)

Y

(None)

Z

(None)

Approach LotusScript Enumeration Values

Approach scripts use the following enumeration values to specify the various types of display objects available in Approach.

Note When you use an enumeration value in a script, you MUST prepend a dollar sign ($) to the enumeration to distinguish the enumerator from a variable.

Object positions	LtsPositionTopLeft
	LtsPositionTop
	LtsPositionTopRight
	LtsPositionLeft
	LtsPositionCenter
	LtsPositionRight
	LtsPositionBottomLeft
	LtsPositionBottom
	LtsPositionBottomRight
	LtsPositionNone
Border relief styles	LtsReliefNone
	LtsReliefRaised
	LtsReliefLowered
Picture Display options	LtsPictureDisplayCrop
	LtsPictureDisplayShrink
Line object orientation	LtsOrientationNegSlope
	LtsOrientationPosSlope
Line styles	LtsLineStyleNone
	LtsLineStyleSolid
	LtsLineStyleDot
	LtsLineStyleShortDash
	LtsLineStyleMediumDash
	LtsLineStyleLineLongDash
	LtsLineStyleMediumShortDash
	LtsLineStyleMediumShortShortDash
	LtsLineStyleDashDot
	LtsLineStyleDashDotDot
	LtsLineStyleDouble
Line spacing	LtsLineSpacingSingle
	LtsLineSpacingSingleAndHalf
	LtsLineSpacingDouble
Display object alignment	LtsAlignmentLeft
	LtsAlignmentRight
	LtsAlignmentHorizCenter
	LtsAlignmentSmart
	LtsAlignmentJustify
	LtsAlignmentTop
	LtsAlignmentVertCenter
	LtsAlignmentBottom
Line widths	AprHairline

	AprHalfPoint
	Apr1Point
	Apr2Point
	Apr3Point
	Apr6Point
	Apr12Point
Border patterns	LtsBorderPatternNone
	LtsBorderPatternSolid
	LtsBorderPatternDashDot
	LtsBorderPatternDashDotDot
	LtsBorderPatternLongDash
	LtsBorderPatternDashed
	LtsBorderPatternDot
	LtsBorderPatternRaised
	LtsBorderPatternLowered
	LtsBorderPatternDouble
	LtsBorderStyleNone
View object types	AprPicture
	AprPicturePlus
	AprButton
	AprLineObject
	AprEllipse
	AprRectangle
	AprRoundRect
	AprOleObject
	AprFieldBox
	AprListBox
	AprCheckBox
	AprDropDownBox
	AprRadioButton
	AprTextBox
View types	AprForm
	AprReport
	AprChart
	AprCrosstab
	AprWorksheet
	AprEnvelope
	AprMailingLabel
	AprFormLetter
Panel types	AprBodyPanel
	AprHeaderFooter
	AprRepeatingPanel
	AprSummaryPanel
	AprSummaryTrailing
	AprSummaryLeading

	AprListField
	AprListUser
Field types	AprFieldNumber
	AprFieldText
	AprFieldPicture
	AprFieldDate
	AprFieldTime
	AprFieldBool
	AprFieldMemo
	AprFieldCalculation
	AprFieldVariable
Summary calculations	AprCalcAverage
	AprCalcCount
	AprCalcSum
	AprCalcMin
	AprCalcMax
	AprCalcStDev
	AprCalcVariance
Chart types	AprChartTypeBar
	AprChartTypeStackBar
	AprChartType100PStackBar
	AprChartTypeHorizBar
	AprChartTypeHorizStackBar
	AprChartType25DBar
	AprChartType25DStackBar
	AprChartType100P25DStackBar
	AprChartType25DHorizBar
	AprChartType25DHorizStackBar
	AprChartTypeLine
	AprChartTypeArea
	AprChartTypeMixed
	AprChartTypePie
	AprChartTypeMultiplePie
	AprChartType25DLine
	AprChartType25DArea
	AprChartType25DMixed
	AprChartType3DPie
	AprChartType3DMultiplePie
	AprChartType3DBar
	AprChartType3DLine
	AprChartType3DArea
	AprChartType3DMixed
	AprChartTypeHLCO
	AprChartTypeRadar
	AprChartTypeXY

AprChartTypeTable
AprChartTypeGANTT
AprChartTypeOrg
AprChartType25DHorBar
AprChartType25DHorStackBar
AprChartTypeSurface
AprChartType3DStackBar
AprChartType100P3DStackBar
AprChartType100PHorizBar
AprChartType100P25DHorizBar
AprChartTypeHLCOJapan
AprChartTypeRadarArea

Approach LotusScript Constants

Approach scripts use the following constants to specify menu commands and colors available in Approach.

Menu command	Constant
About Approach...	IDM_ABOUT
Actual Size	IDM_ACTUAL
Add Column	IDM_ADDCOLUMN
Approach File Info...	IDM_APRINFO
Tile Left-Right	IDM_TILE_LEFT_RIGHT
Tile Top-Bottom	IDM_TILE_TOP_BOTTOM
Ascending Sort	IDM_SORTUP
Browse	IDM_BROWSE
Chart Crosstab	IDM_CHART
Clear	IDM_CLEAR
Close	IDM_CLOSE
Copy	IDM_COPY
Copy to File...	IDM_COPYTOFILE
Create Chart...	IDM_NEWCHART
Create CrossTab...	IDM_NEWXTAB
Create Form...	IDM_NEWFORM
Create Form Letter...	IDM_NEWLETTER
Create Mailing Labels...	IDM_NEWLABELS
Create Object...	IDM_INSOBJECT
Create Control (OLE)...	IDM_INSCONTROL
Create Report...	IDM_NEWREPORT
Create Worksheet...	IDM_NEWWORKSHEET
Customize Menus...	IDM_MENUS
Cut	IDM_CUT
Define Sort...	IDM_SORT
Delete File...	IDM_DELETEFILE
Delete Found Set	IDM_DELETEFOUND
Delete Record	IDM_DELETERECORD
Descending Sort	IDM_SORTDOWN
Design	IDM_DESIGN
Duplicate Record	IDM_DUPLICATE
Edit Column Label	IDM_EDITLABEL
Edit OLE Object...	IDM_EDITOLE
Export Data...	IDM_EXPORT
Exit	IDM_EXIT
Fast Format	IDM_FASTFORMAT
Field Definition...	IDM_DEFINE
Fill Field...	IDM_FILL
Find	IDM_FIND
Find Again	IDM_FINDAGAIN
Find Special...	IDM_FINDSPECIAL
First Record	IDM_FIRST

Help Contents	IDM_HELPINDEX
Help Customer Support	IDM_HELPSUPPORT
Help For Upgraders	IDM_HELPUPGRADE
Help Functions	IDM_HELPFUNCTION
Help How Do I?	IDM_HELPHOWTO
Help Keyboard	IDM_HELPKEYS
Help Search...	IDM_HELPSEARCH
Help Using Help	IDM_HELPHELP
Help Working Together	IDM_HELPWT
Hide Record	IDM_HIDE
Import Data...	IDM_IMPORT
Insert Today's Date	IDM_DATE
Insert Current Time	IDM_TIME
Insert Previous Value	IDM_PREVDATA
Join...	IDM_JOIN
Last Record	IDM_LAST
Links...	IDM_LINKS
Macros...	IDM_MACROS
Named Styles...	IDM_STYLES
New...	IDM_NEW
New Record	IDM_NEWREC
Next Record	IDM_NEXT
Notes New Notes Replica...	IDM_NEWREPLICA
Notes Replicate with Notes Server...	IDM_REPLICATE
Open...	IDM_OPEN
Preferences...	IDM_OPTIONS
Paste	IDM_PASTE
Paste from File...	IDM_PASTEFILE
Paste Special...	IDM_PASTELINK
Preview	IDM_PREVIEW
Previous Record	IDM_PREV
Print...	IDM_PRINT
Print Setup...	IDM_PRINTSETUP
Refresh	IDM_REFRESH
Save As...	IDM_SAVEAS
Save Approach File	IDM_SAVE
Select All	IDM_SELECTALL
Select Cells Only	IDM_SELECTCELLS
Select Header Only	IDM_SELECTLABEL
Send Mail...	IDM_SENDMAIL
Show All	IDM_SHOWALL
Show Data	IDM_SHOWDATA
Show SmartIcons	IDM_SHOWICONS
Show Status Bar	IDM_SHOWSTATUS
Show View Tabs	IDM_SHOWTABS

SmartIcons...	IDM_SMARTICONS
Spell Check...	IDM_SPELLCHECK
Object Properties	IDM_INFOBOX
View Properties	IDM_INFOBOX_VIEW
Summarize Columns	IDM_ADDROW
Summarize Rows	IDM_ADDCOLUMN
Undo	IDM_UNDO
Zoom In	IDM_ZOOMIN
Zoom Out	IDM_ZOOMOUT
Replace	IDM_REPLACE

Color constants	**RGB contributions**
COLOR_WHITE	0x02FFFFFFL
COLOR_VANILLA	0x02FFEFCEL
COLOR_PARCHMENT	0x02FFFFC2L
COLOR_IVORY	0x02FFFFD0L
COLOR_PALE_GREEN	0x02E0FFBFL
COLOR_SEA_MIST	0x02E0FFDFL
COLOR_ICE_BLUE	0x02E0FFFFL
COLOR_POWDER_BLUE	0x02C2EFFFL
COLOR_ARCTIC_BLUE	0x02E0F1FFL
COLOR_LILAC_MIST	0x02E0E0FFL
COLOR_PURPLE_WASH	0x02E8E0FFL
COLOR_VIOLET_FROST	0x02F1E0FFL
COLOR_SEASHELL	0x02FFE0FFL
COLOR_ROSE_PEARL	0x02FFE0F5L
COLOR_PALE_CHERRY	0x02FFE0E6L
COLOR_WHITE	0x02FFFFFFL
COLOR_BLUSH	0x02FFE1DCL
COLOR_SAND	0x02FFE1B0L
COLOR_LIGHT_YELLOW	0x02FFFF80L
COLOR_HONEYDEW	0x02F1F1B4L
COLOR_CELERY	0x02C2FF91L
COLOR_PALE_AQUA	0x02C1FFD5L
COLOR_PALE_BLUE	0x02C1FFFFL
COLOR_CRYSTAL_BLUE	0x02A1E2FFL
COLOR_LT_CORNFLOWER	0x02C0E1FFL
COLOR_PALE_LAVENDER	0x02BFBFFFL
COLOR_GRAPE_FIZZ	0x02D2BFFFL
COLOR_PALE_PLUM	0x02E1BFFFL
COLOR_PALE_PINK	0x02FFC1FDL
COLOR_PALE_ROSE	0x02FFC0E4L
COLOR_ROSE_QUARTZ	0x02FFC0CEL

COLOR_3_GRAY	0x02F7F7F7L
COLOR_RED_SAND	0x02FFC0B6L
COLOR_BUFF	0x02FFC281L
COLOR_LEMON	0x02FFFF35L
COLOR_PALE_LEMON_LIME	0x02F1F180L
COLOR_MINT_GREEN	0x0280FF80L
COLOR_PASTEL_GREEN	0x0282FFCAL
COLOR_PASTEL_BLUE	0x0280FFFFL
COLOR_SAPPHIRE	0x0282E0FFL
COLOR_CORNFLOWER	0x0282C0FFL
COLOR_LIGHT_LAVENDER	0x029F9FFFL
COLOR_PALE_PURPLE	0x02C29FFFL
COLOR_LIGHT_ORCHID	0x02E29FFFL
COLOR_PINK_ORCHID	0x02FF9FFFL
COLOR_APPLE_BLOSSOM	0x02FF9FCFL
COLOR_PINK_CORAL	0x02FF9FA9L
COLOR_6_GRAY	0x02EFEFEFL
COLOR_LIGHT_SALMON	0x02FF9F9FL
COLOR_LIGHT_PEACH	0x02FF9F71L
COLOR_YELLOW	0x02FFFF00L
COLOR_AVOCADO	0x02E0E074L
COLOR_LEAF_GREEN	0x0241FF32L
COLOR_LIGHT_AQUA	0x0242FFC7L
COLOR_LT_TURQUOISE	0x0242FFFFL
COLOR_LIGHT_CERULEAN	0x0200BFFFL
COLOR_AZURE	0x025291EFL
COLOR_LAVENDER	0x028080FFL
COLOR_LIGHT_PURPLE	0x02C082FFL
COLOR_DUSTY_VIOLET	0x02E081FFL
COLOR_PINK	0x02FF7FFFL
COLOR_PASTEL_PINK	0x02FF82C2L
COLOR_PASTEL_RED	0x02FF82A0L
COLOR_12_GRAY	0x02E1E1E1L
COLOR_SALMON	0x02FF8080L
COLOR_PEACH	0x02FF8141L
COLOR_MUSTARD	0x02FFE118L
COLOR_LEMON_LIME	0x02E1E140L
COLOR_NEON_GREEN	0x0200FF00L
COLOR_AQUA	0x0200FFB2L
COLOR_TURQUOISE	0x0200FFFFL
COLOR_CERULEAN	0x0200A1E0L
COLOR_WEDGEWOOD	0x022181FFL

COLOR_HEATHER	0x026181FFL
COLOR_PURPLE_HAZE	0x02A160FFL
COLOR_ORCHID	0x02C062FFL
COLOR_FLAMINGO	0x02FF5FFFL
COLOR_CHERRY_PINK	0x02FF60AFL
COLOR_RED_CORAL	0x02FF6088L
COLOR_18_GRAY	0x02D2D2D2L
COLOR_DARK_SALMON	0x02FF4040L
COLOR_DARK_PEACH	0x02FF421EL
COLOR_GOLD	0x02FFBF18L
COLOR_YELLOW_GREEN	0x02E1E100L
COLOR_LIGHT_GREEN	0x0200E100L
COLOR_CARIBBEAN	0x0200E1ADL
COLOR_DK_PASTEL_BLUE	0x0200E0E0L
COLOR_DARK_CERULEAN	0x020082BFL
COLOR_MANGANESE_BLUE	0x020080FFL
COLOR_LILAC	0x024181FFL
COLOR_PURPLE	0x028242FFL
COLOR_LT_RED_VIOLET	0x02C140FFL
COLOR_LIGHT_MAGENTA	0x02FF42F9L
COLOR_ROSE	0x02FF40A0L
COLOR_CARNATION_PINK	0x02FF4070L
COLOR_25_GRAY	0x02C0C0C0L
COLOR_WATERMELON	0x02FF1F35L
COLOR_TANGERINE	0x02FF1F10L
COLOR_ORANGE	0x02FF8100L
COLOR_CHARTREUSE	0x02BFBF00L
COLOR_GREEN	0x0200C200L
COLOR_TEAL	0x0200C196L
COLOR_DARK_TURQUOISE	0x0200C1C2L
COLOR_LT_SLATE_BLUE	0x024181C0L
COLOR_MEDIUM_BLUE	0x020062E1L
COLOR_DARK_LILAC	0x024141FFL
COLOR_ROYAL_PURPLE	0x024200FFL
COLOR_FUCHSIA	0x02C200FFL
COLOR_CONFETTI_PINK	0x02FF22FFL
COLOR_PALE_BURGANDY	0x02F52B97L
COLOR_STRAWBERRY	0x02FF2259L
COLOR_30_GRAY	0x02B2B2B2L
COLOR_ROUGE	0x02E01F25L
COLOR_BURNT_ORANGE	0x02E12000L
COLOR_DARK_ORANGE	0x02E26200L

COLOR_LIGHT_OLIVE	0x02A1A100L
COLOR_KELLY_GREEN	0x0200A000L
COLOR_SEA_GREEN	0x02009F82L
COLOR_AZTEC_BLUE	0x02008080L
COLOR_DUSTY_BLUE	0x020060A0L
COLOR_BLUEBERRY	0x020041C2L
COLOR_VIOLET	0x020021BFL
COLOR_DEEP_PURPLE	0x024100C2L
COLOR_RED_VIOLET	0x028100FFL
COLOR_HOT_PINK	0x02FF00FFL
COLOR_DARK_ROSE	0x02FF0080L
COLOR_POPPY_RED	0x02FF0041L
COLOR_36_GRAY	0x02A2A2A2L
COLOR_CRIMSON	0x02C20000L
COLOR_RED	0x02FF0000L
COLOR_LIGHT_BROWN	0x02BF4100L
COLOR_OLIVE	0x02808000L
COLOR_DARK_GREEN	0x02008000L
COLOR_DARK_TEAL	0x02008250L
COLOR_SPRUCE	0x02006062L
COLOR_SLATE_BLUE	0x02004080L
COLOR_NAVY_BLUE	0x02001FE2L
COLOR_BLUE_VIOLET	0x024040C2L
COLOR_AMETHYST	0x024000A2L
COLOR_DK_RED_VIOLET	0x026000A1L
COLOR_MAGENTA	0x02E000E0L
COLOR_LIGHT_BURGUNDY	0x02DF007FL
COLOR_CHERRY_RED	0x02C20041L
COLOR_44_GRAY	0x028F8F8FL
COLOR_DARK_CRIMSON	0x02A00000L
COLOR_DARK_RED	0x02E10000L
COLOR_HAZELNUT	0x02A13F00L
COLOR_DARK_OLIVE	0x02626200L
COLOR_EMERALD	0x02006000L
COLOR_MALACHITE	0x0200603CL
COLOR_DARK_SPRUCE	0x02004041L
COLOR_STEEL_BLUE	0x02002F80L
COLOR_BLUE	0x020000FFL
COLOR_IRIS	0x022020A0L
COLOR_GRAPE	0x022200A1L
COLOR_PLUM	0x02400080L
COLOR_DARK_MAGENTA	0x02A1009FL

COLOR_BURGANDY	0x02C0007FL
COLOR_CRANBERRY	0x029F000FL
COLOR_50_GRAY	0x02808080L
COLOR_MAHOGANY	0x02600000L
COLOR_BRICK	0x02C21212L
COLOR_DARK_BROWN	0x02824200L
COLOR_DEEP_OLIVE	0x02424200L
COLOR_DARK_EMERALD	0x02004200L
COLOR_EVERGREEN	0x02004023L
COLOR_BALTIC_BLUE	0x0200323FL
COLOR_BLUE_DENIM	0x02002060L
COLOR_COBALT_BLUE	0x020020C2L
COLOR_DARK_IRIS	0x022222C0L
COLOR_MIDNIGHT	0x02000080L
COLOR_DARK_PLUM	0x021F007FL
COLOR_PLUM_RED	0x02800080L
COLOR_DARK_BURGANDY	0x02820040L
COLOR_SCARLET	0x02800000L
COLOR_63_GRAY	0x025F5F5FL
COLOR_CHESTNUT	0x02400000L
COLOR_TERRA_COTTA	0x02A11F12L
COLOR_UMBER	0x02604200L
COLOR_AMAZON	0x02212100L
COLOR_PEACOCK_GREEN	0x02002100L
COLOR_PINE	0x0200201FL
COLOR_SEAL_BLUE	0x02002041L
COLOR_DK_SLATE_BLUE	0x0200204FL
COLOR_ROYAL_BLUE	0x020000E0L
COLOR_LAPIS	0x020000A1L
COLOR_DARK_GRAPE	0x02000061L
COLOR_AUBERGINE	0x021F0062L
COLOR_DARK_PLUM_RED	0x0240005FL
COLOR_RASPBERRY	0x02620042L
COLOR_DEEP_SCARLET	0x02620012L
COLOR_69_GRAY	0x024F4F4FL
COLOR_BURNT_SIENNA	0x02200000L
COLOR_MILK_CHOCOLATE	0x02622100L
COLOR_BURNT_UMBER	0x02412000L
COLOR_DEEP_AVOCADO	0x02101000L
COLOR_DEEP_FOREST	0x02001000L
COLOR_DARK_PINE	0x0200120CL
COLOR_DK_METALIC_BLUE	0x0200121FL

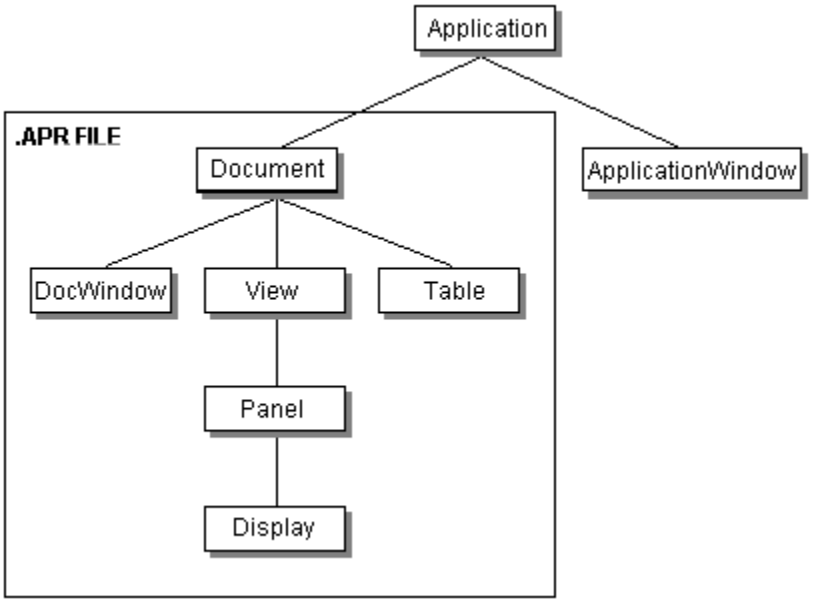
COLOR_AIR_FORCE_BLUE	0x02001040L
COLOR_ULTRAMARINE	0x020000C2L
COLOR_PRUSSIAN_BLUE	0x020000AFL
COLOR_RAISIN	0x0200004FL
COLOR_EGGPLANT	0x02000040L
COLOR_BOYSENBERRY	0x02200042L
COLOR_BORDEAUX	0x02400040L
COLOR_RUBY	0x02410012L
COLOR_75_GRAY	0x02404040L
COLOR_RED_GRAY	0x02D0B1A1L
COLOR_TAN	0x02E0A175L
COLOR_KHAKI	0x02D2B06AL
COLOR_PUTTY	0x02C0C27CL
COLOR_BAMBOO_GREEN	0x0282C168L
COLOR_GREEN_GRAY	0x0281C097L
COLOR_BALTIC_GRAY	0x027FC2BCL
COLOR_BLUE_GRAY	0x0271B2CFL
COLOR_RAIN_CLOUD	0x02B1B1D2L
COLOR_LILAC_GRAY	0x029F9FE0L
COLOR_LT_PURPLE_GRAY	0x02C0A1E0L
COLOR_LIGHT_MAUVE	0x02E29FDEL
COLOR_LT_PLUM_GRAY	0x02EF91EBL
COLOR_LT_BURGANDY_GRAY	0x02E29FC8L
COLOR_ROSE_GRAY	0x02F18FBCL
COLOR_82_GRAY	0x022F2F2FL
COLOR_DARK_RED_GRAY	0x027F604FL
COLOR_DARK_TAN	0x02A16252L
COLOR_SAFARI	0x02806210L
COLOR_OLIVE_GRAY	0x0282823FL
COLOR_JADE	0x023F621FL
COLOR_DK_GREEN_GRAY	0x023C613EL
COLOR_SPRUCE_GRAY	0x0237605EL
COLOR_DK_BLUE_GRAY	0x02104160L
COLOR_ATLANTIC_GRAY	0x02424282L
COLOR_DK_LILAC_GRAY	0x026260A1L
COLOR_PURPLE_GRAY	0x02624181L
COLOR_MAUVE	0x02603181L
COLOR_PLUM_GRAY	0x02602162L
COLOR_BURGUNDY_GRAY	0x02622152L
COLOR_DARK_ROSE_GRAY	0x02813F62L
COLOR_BLACK	0x02000000L
COLOR_TRANSPARENT	0xFFFFFFFFL

0xFFFFFFFFL

Approach LotusScript Class Hierarchy

Understanding which classes are contained by other classes, and therefore which objects are contained by other objects, helps you understand the syntax to use in a script when you try to access an object or change one of its properties. One of the advantages of containment is that it lets you access any object by traversing the containment hierarchy that connects objects with other objects.

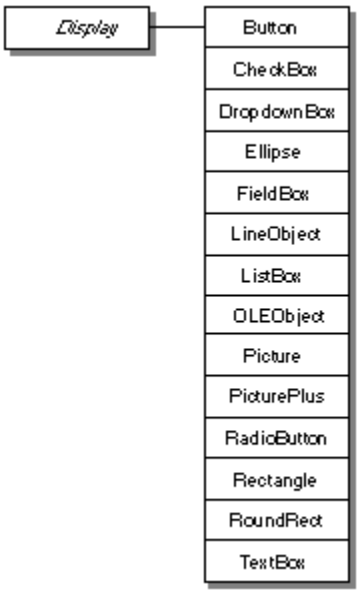
The following diagram illustrates the containment relationships of classes and objects.

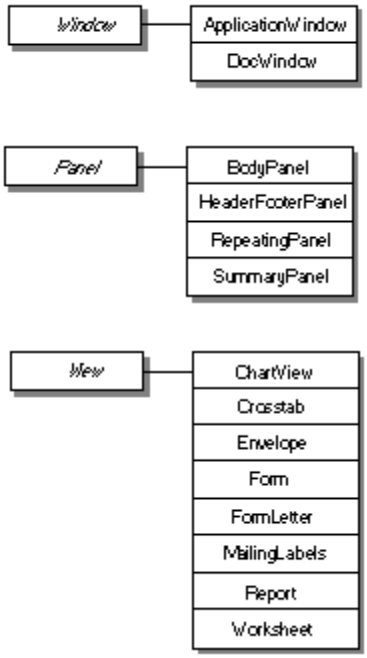


Approach provides some classes, called abstract classes, that exist only to create other classes, called derived classes. You cannot create an instance (object) of an abstract class.

A derived class, also called a subclass, inherits the members (methods and properties) of the class it derives from, called its base class.

The following diagrams show the Approach class hierarchy, including the abstract classes (in italics) and the derived classes that inherit from them.




Window
ApplicationWindow
DocWindow
Panel
BodyPanel
HeaderFooterPanel
RepeatingPanel
SummaryPanel
View
ChartView
Crosstab
Envelope
Form
FormLetter
MailingLabels
Report
Worksheet