EXTRA!

OLE Automation
Programmer's Reference

Manual 229055 September 1996

For help with this product, contact Attachmate's Customer Support. (Refer to the back cover of this manual for the phone number.)

Table of Contents

This manual describes the objects, methods, and properties of *EXTRA!*, with which you can programmatically access the product using OLE Automation. The manual introduces you to key OLE Automation concepts as they relate to *EXTRA!*, and serves as a reference to the product's objects, properties, and methods.

Syntax Conventions

This manual uses the following conventions:

Convention	Description
Session, Activate, Col, Set	Bolded words are reserved, such as object, property, and method names. Type these words as they appear in Help.
object, rc, startRow	Italicized words indicate placeholders for object or variable names, text, or values that you supply.
{**3270** | **VT** | **5250**}	Braces and vertical bars indicate that you must enter one of the items.
[,*Page*]	Brackets indicate that a parameter is optional.
...	Ellipses indicate that the previous parameter can be repeated.
_	The underscore is a line-continuation character used in code examples. Placed at the end of a line, the character indicates that code continued from one line to the next is part of the same logical line.

Note: The line-continuation character is more than notational; you can include it in your code to break up lines — provided that your macro language supports it. *EXTRA!* Basic, Visual Basic 4.0, and Visual Basic, Applications Edition support the underscore as line-continuation character. Versions of Visual Basic prior to 4.0 do not.

Related Documentation

In addition to the *OLE Automation Programmer's Reference*, *EXTRA!* includes the following documentation:

Manuals

- *EXTRA! User's Guid*e, which includes information about 3270, 5250 and VT file transfer, customizing sessions, printing, and macros.
- *EXTRA! Basic Language Reference,* which provides a language overview and summary, as well as an alphabetical reference to functions and statements.

These manuals can be purchased separately or as a set.

Online Help

EXTRA! includes a comprehensive set of online Help, including help for the *EXTRA!* Basic macro language and OLE Automation. If you use *EXTRA!* Basic to access the *EXTRA!* objects, you will find the context-sensitive help in the Macro Editor especially useful. You can access this help in the following ways:

- Display a detailed online Help topic about a particular *EXTRA!* Basic or OLE Automation element. Highlight the function, statement, or call in the Editor workspace and press F1. The Help Index is displayed pointing to the highest-level entry pertaining to the highlighted language item or object. Choose an appropriate sub-entry, if desired, and then choose the Display button.

  –or–

  From the Macro Editor Functions and Objects browser, highlight an *EXTRA!* Basic or OLE Automation element and choose the question mark button.
- Click the Help button that is located in each Macro Editor dialog box,
- Activate "What's This" mode by selecting an item in a Macro Editor dialog box and then clicking the right mouse button, or
- Click on the Help Mode toolbar button, then click on an element in the Macro Editor user interface.

Let Us Know How We're Doing

Please fill out the Customer Documentation Feedback Card that comes with your *EXTRA!* software package to let us know what you think about the documentation. Your comments are valuable to us.

CHAPTER

Introduction

EXTRA! supports OLE Automation, a Microsoft Windows standard for inter-program communications. As an OLE Automation server, *EXTRA!* exposes programmable objects like Session, Screen, and Toolbar, whose built-in functionality can be accessed by user-written applications. For example, you could write a PC application that transfers files between an IBM mainframe, AS/400, and VAX. You would not have to write the file transfer code, but merely invoke the file transfer functionality inherent in 3270, 5250, and VT Session objects. To access automation-enabled objects, you must use a macro or programming language that supports OLE Automation, such as the *EXTRA!* Basic language or Microsoft's Visual Basic 3.0 or later.

You control objects through their properties and methods, using the dot syntax of *object.property* and *object.method.* This chapter introduces you to the OLE Automation concepts of objects, properties, and methods.

Overview of Objects

An object is one component of an application, which you can access and control through the object's attributes and commands, referred to as properties and methods. *EXTRA!* consists of the following objects:

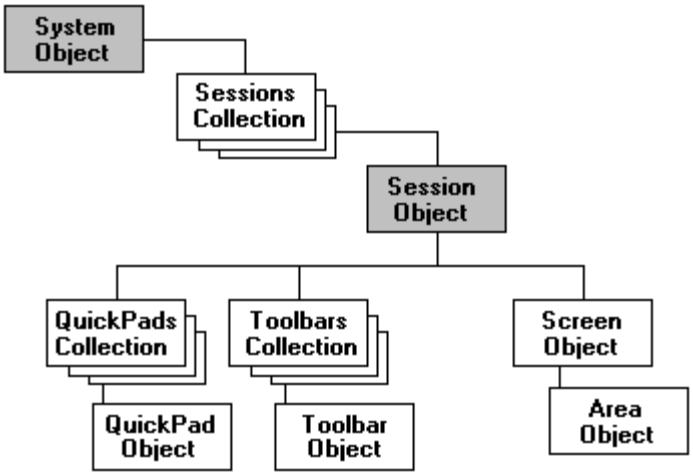
- System — Top-level object, providing access to all objects in *EXTRA!*.
- Sessions Collection — Manages open session objects.
- Session — Provides access to host data and *EXTRA!* functionality.
- Screen — Provides access to the contents of the host presentation space, an area in PC memory that stores the screen data of a display session.
- Area — Provides access to a defined area of the screen.
- QuickPads Collection — Manages available QuickPad objects. A QuickPad is a configurable secondary window from which you can execute *EXTRA!* commands, macros, and host functions.
- QuickPad — Provides access to a specific QuickPad to be used with a session.
- Toolbars Collection — Manages available Toolbar objects. A Toolbar is a configurable bar with buttons from which you can execute *EXTRA!* commands, macros, and host functions.
- Toolbar — Provides access to a specific Toolbar to be used with a session.

Some of the *EXTRA!* objects are the same as the visual objects that appear when you run the application manually, like a session or a toolbar. Other objects cannot be displayed. For example, an Area object, which defines a section of a screen, is not visible.

The Object Model

Objects have a hierarchical relationship. As shown in the object model below, the System object is at the top level; it must be retrieved before any other objects. Similarly, a Session object must be retrieved before a Screen object. Subordinate objects like Screen are returned by a property or method of a different object type, like Session. Objects returned in this way are referred to as sub-objects.

The shaded objects represent entry points into *EXTRA!* Objects. You can start the System object with the CreateObject function. A more direct way to start a session, however, is by retrieving a Session object from a file with the GetObject function.



Collection Objects

The object model also shows collection objects — Sessions, QuickPads, and Toolbars. An object that manages a set of related objects is a collection. For example, the Sessions object manages individual Session objects; the Toolbars and QuickPad objects manage Toolbar and QuickPad objects. Objects within a collection are sometimes referred to as elements.

A collection, which consists of zero or more elements, includes a Count property that returns the current number of elements For example, the expression **Sessions.Count** returns the number of Session objects that are currently open.

To retrieve an individual element within a collection, you must identify the element with the collection's Item method. The Item method takes a numeric argument that represents the element's ordinal position within the collection. The Item property can also take a string argument that specifies an element's name. The following expressions show two ways to return Session 3, the third element in the Sessions collection object.

```
Sessions.Item("Session3")
```

–or–

```
Sessions.Item(3)
```

Note: Because Item is the default method of collection objects, you do not have to explicitly specify the method. Session 3 in the collection can also be referenced with **Sessions(3).Close**.

Like any returned objects, objects returned by properties and methods of collections can be assigned to object variables. For example:

```
Set Ses3 = Sessions(3)
```

A collection object makes it easy to iterate (perform repetitive actions) on all of the objects within the collection. Using Microsoft's Visual Basic, the For...Each statement is one way to iterate through the elements of a collection. For example, the following For...Next statement loops through all open sessions and minimizes their windows.

```
Dim Ses As Object
For Each Ses in extra.Sessions
       Ses.WindowState = xMINIMIZED
Next
```

Retrieving and Referencing Objects

You can retrieve objects with functions, properties, and methods. To reference a retrieved object, you must assign it to an object variable.

Retrieving Objects

The most direct way to retrieve a Session object is with the GetObject function, which retrieves a reference to an object from a file. For example, the following expression retrieves a Session object from the file Session1.EDP.

```
GetObject("Session1.EDP")
```

Although this expression does not include a reference to the System object, which is at the top level of the object hierarchy, it nonetheless creates a System object before retrieving the Session object.

Each *EXTRA!* object has properties that return references to other objects. For example, the expression **System.Sessions** returns a reference to a Sessions collection object. The expression **Session.Screen** returns a reference to a Screen object.

Some objects also include methods that return references to objects. For example, the expression **Screen.Area(1,1,10,80)** returns an Area object corresponding with an area of the screen.

Referencing Objects

To control a retrieved object, you must assign an object reference to an object variable. First, declare an object variable with a Dim, Static, or Global statement. Next, use the Set statement to assign to the object variable a reference to the object.

Note: The Set statement assigns an object reference to a variable, not a copy of the object. Therefore, more than one object variable can refer to the same object. Any change to an object affects all variables that reference that object.

In the following lines, two object variables are declared. In the first Set statement, the GetObject function returns a reference to a Session object, assigned to the object variables Ses1. In the second Set statement, the Screen property of Ses1 is used. A reference to a Screen object is returned and assigned to the object variable SessionScreen.

```
Dim Ses1 As Object, SessionScreen As Object
Set Ses1 = GetObject("Session1.EDP")
Set SessionScreen = Ses1.Screen
```

Object variables are necessary for referencing objects. However, for objects that must be retrieved but not referenced, it is more efficient to use a compound statement rather than assigning an object to a variable.

Compound Statements

A compound statement includes a two or more expressions, each of which returns an object. Compound statements are useful when you have to traverse the object hierarchy to reference an object. For example, the following compound statement returns a reference to a Screen object.

```
Set SessionScreen =
Extra.Sessions(1).Screen.Area(1,1,10,80)
```

This statement is evaluated as follows:

Expression	Object Returned
Extra.Sessions	Sessions collection object.
Sessions(1) (equivalent to Sessions.Item(1))	The first session in the collection.
Screen property of the first session	Screen object.
The Area method of the returned Screen object	An Area object that defines a screen segment from row 1, column 1 to row 10, column 80.

Because only the Screen object has to be referenced, the use of the compound statement is far more efficient than assigning multiple object references to multiple object variables. Compare the compound statement to the equivalent set of statements below.

```
Dim SessionsObj As Object
Dim SessionObj As Object
Dim ScreenObj As Object
Dim AreaObj As Object
Set SessionsObj = extra.Sessions
Set SessionObj = SessionsObj(1)
Set ScreenObj = SessionObj.Screen
Set AreaObj = ScreenObj.Area(1,1,10,80)
```

Overview of Properties

A property is a unique attribute of an object that you can return or set. To refer to a property, use the following dot syntax:

```
object.property
```

Some properties take arguments, which you enclose within parentheses as follows:

```
object.property (arg1,arg2,...)
```

By setting the values of an object's properties, you change the object's characteristics. For example, you can use the ColorScheme property of the Session object to change a session's color scheme association. The following statement sets the color scheme to "Blue Sky."

```
Set Session.ColorScheme = "Blue Sky"
```

You can determine the current color scheme of the session object as well as set it. The following statement returns the current value of the ColorScheme property and displays it.

```
MsgBox Session.ColorScheme
```

Properties that you can both set and return, like ColorScheme, are read-write properties. Properties whose values can only be returned are read-only properties. Most of the System object's properties are read-only. For example, you can return the values of the ActiveSession, Version, and Application properties, but you cannot change their values.

Properties that Return Objects

The value of most properties is either an integer, a character string, or a Boolean value. However, some properties return an object. In the statement below, **Ses1.Screen** returns a reference to a Screen object. Ses1 is an object variable that references a Session object; Screen is a Session object property. The Set statement assigns the returned Screen object reference to the object variable SessionScreen.

```
Set SessionScreen = Ses1.Screen
```

In the next statement, **Extra.ActiveSession.Close**, the Session object that currently has the focus is closed. Extra is an object variable that references the System object; ActiveSession is a property of the System object that returns the Session object that currently has the focus. The Close part of the expression is a method of the Session object.

```
Extra.ActiveSession.Close
```

Default Properties

Every object has a default property, that is, a property that does not have to be explicitly stated in an expression. For example, the Name property is the default for the Session object. The following two statements both return the name value for a Session object referred to as Ses1.

```
SesName = Ses1.Name
```

–or–

```
SesName = Ses1
```

Overview of Methods

A method is a command that directs an object to perform an action. For example, the JumpNext method of the Sessions object transfers focus from the currently active session to the next session. The Copy method of the Screen object copies the selected screen area to the Clipboard.

To refer to a method, use the following dot syntax:

```
object.method
```

Some methods take arguments:

```
object.method arg1,arg2,...
```

Some methods return values as well:

```
rc = object.method (arg1,arg2,...)
```

Note the syntax difference in the last two statements: when the return value is saved to a variable, the arguments are enclosed in parentheses. Even if there is no argument list, include parentheses:

```
rc = object.method ( )
```

Every object has a default method, that is, a method that does not have to be explicitly stated in an expression. For example, the Item method is the default method for collection objects. Each of the following two statements closes the third session in the Sessions collection:

```
Sessions.Item(3).Close
```

–or–

```
Sessions(3).Close
```

CHAPTER

2

Alphabetical Reference to Objects, Properties, and Methods

Activate Method

Applies To Objects

Session

Description

Makes the specified session the active window.

Syntax

object.**Activate**

Element	Application
object	The Session object.

Activate Method Example

Making each open session the active window, this example displays the name of the session that is currently active.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes one or more open sessions
    Set Sess = Sys.ActiveSession
    For i=1 to Sys.Sessions.Count
        Set Sess=Sys.Sessions.Item(i)
        MsgBox "Activating session " + Sess.Name + "."
        Sess.Activate
    Next
End Sub
```

ActiveSession Property

Applies To Objects

System

Description

Returns the currently active Session object. Read-only.

Syntax

object.ActiveSession

Element	Description
object	The System object.

Comments

You can assign the returned object to a variable or property with the Set statement.

ActiveSession Property Example

After initializing the object variable sess with a reference to the active session, this example displays the name of the session.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    SessionName$ = Sess.Name
    MsgBox "The active session is " +SessionName$+"."
End Sub
```

Application Property

Applies To Objects

Area, QuickPad, QuickPads. Screen, Session, Sessions,System, Toolbar, Toolbars

Description

Returns the System object. Read-only.

Syntax

object.**Application**

Element	Description
object	One of the above-listed objects.

Application Property Example

This example shows how the System object can be returned from any object.

```
Sub Main()
    Dim Sys As Object, AllSess As Object, Sess As Object
    Dim MyScreen As Object, MyArea As Object
    Dim TBars As Object, QPads As Object, MyTBar As Object, _
    MyQPads As Object

    Set Sys = CreateObject("EXTRA.System")
    Set AllSess = Sys.Sessions
' Assumes an open session
    Set Sess = Sys.ActiveSession

    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(1, 1, 10, 10, , ,)
    Set TBars = Sess.Toolbars
    Set QPads = Sess.QuickPads
```

```
    Set MyTBar = TBars(1)
    Set MyQPad = QPads(1)

    '  All these return the same result.
    MsgBox "System.Application = " + Sys.Application.Name
    MsgBox "Sessions.Application = " + AllSess.Application.Name
    MsgBox "Session.Application = " + Sess.Application
    MsgBox "Screen.Application = " + MyScreen.Application.Name
    MsgBox "Area.Application = " + MyArea.Application.Name
    MsgBox "Toolbars.Application = " + TBars.Application.Name
    MsgBox "QuickPads.Application = " + QPads.Application.Name
    MsgBox "ToolBar.Application = " + MyTBar.Application.Name
    MsgBox "QuickPad.Application = " + MyQPad.Application.Name
End Sub
```

Area Method

Applies To Objects

Screen

Description

Returns an Area object with the defined coordinates.

Syntax

```
Set rc = object.Area(StartRow, _
StartCol,EndRow,EndCol[,Page][,Type])
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Screen object.
StartRow	The row where the selection begins.
StartCol	The column where the selection begins.
EndRow	The row where the selection ends.
EndCol	The column where the selection ends.
Page	VT session only—the screen page. **Note:** This parameter is ignored for release 6.0 of *EXTRA!*.
Type	Determines how the coordinates of the Area object (top, left, bottom, right) are interpreted when the object is selected.

Comments

The Area method requires six parameters. The Page and Type parameters are optional in the sense that you do not have to specify values; however, if you don't specify values for these parameters, they must be represented by placeholders, namely commas. In the following statement, the Area method includes six parameters, with the last two being the comma placeholders for the Page and Type parameters.

```
Set MyArea = MyScreen.Area(1,1,80,24 , ,)
```

The Type parameter accepts the following values.

Value	Meaning
3 or xBLOCK	The Area is selected as a rectangular range of characters. There is no line wrapping. This is the default.
2 or xSTREAM	The Area is selected as a continuous stream of characters, from the top left coordinate to the bottom right coordinate. If the Area consists of more than one line, then the selection wraps to the right of each line.
1 or xPOINT	The Area is selected as a single point. The bottom and right coordinates are ignored. Not supported by VT session.
0 or xNONE	Invalidates selection of the Area.

Area Method Example

This example returns an Area object and displays the text in the object.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open sessions
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(1,1,MyScreen.Rows,1 , ,)
    MsgBox "The text in the first column is " + MyArea.Value + "."
End Sub
```

Area Object

Description

Provides access to a defined area of the screen.

Properties	Methods
Application	Copy
Bottom	Cut
Left	Delete
Parent	Paste
Right	Select
Top	
Type	
Value	

Comments

You can create Area objects using methods and properties of the Screen object, such as Area and Selection.

Using the methods and properties of an Area object, you can read from or write to the presentation space, an area in PC memory that stores the screen data of a terminal session. A presentation space includes data from the status line (called Operator Information Area in 3270 sessions). Each presentation space position stores a screen character.

To access the presentation space, you must know the exact number of rows and columns that an emulated terminal provides. For example, if a session emulates a terminal supporting 24 rows by 80 columns, you can reference presentation space positions from row 1, column 1 to row 24, column 80. For VT sessions, you can also specify the page.

Bottom Property

Applies To Objects

Area

Description

Returns or sets the ending row of the Area object. Read-write.

Syntax

object.**Bottom**

Element	Description
object	The Area object.

Comments

You can set the Bottom property to shrink or expand an Area object. For example, if you initially defined an area with a bottom row of ten, you can set the bottom row to fifteen, thereby expanding the size of the area.

You can also adjust the size of the Area object with the Top, Left, and Right properties.

Bottom Property Example

This example uses Top and Bottom properties with an Area object to narrow a selection on the screen. Then the example uses the Top property with a Session object to move the session around on the screen.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
    ' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
```

```
' This illustrates the Top and Bottom properties for an Area
   ' object. For demonstration purposes, the Select method is used,
   ' but it is not required for setting Top and Bottom properties.
   MsgBox "Press to demonstrate the Top and Bottom properties _
   for Area objects."
   Set MyArea = MyScreen.Area(1, 1, MyScreen.Rows, _
   MyScreen.Cols,,3)
   MyArea.Select
   For i = 1 to Int(MyScreen.Rows/2)
     MyArea.Top = MyArea.Top + 1
     MyArea.Select
     MyArea.Bottom = MyArea.Bottom - 1
     MyArea.Select
   Next

   ' This demonstrates the Top property for a Session object.
   MsgBox "Press to demonstrate the Top property for Session _
   objects."
   Sess.Top = 50
   MsgBox "The session top is now at 50.  Press to move session _
   top to 1"
   Sess.Top = 1
   MsgBox "The session top is now at 1.  Press to move session _
   top to 100"
   Sess.Top = 100
End Sub
```

Close Method

Applies To Objects

Session

Description

Closes the session.

Syntax

object.**Close**

Element	Application
object	The Session object.

Close Method Example

This example closes the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    MsgBox "Press to close session."
    Sess.Close
End Sub
```

CloseAll Method

Applies To Objects

Sessions

Description

Closes all active sessions.

Syntax

object.**CloseAll**

Element	Description
object	The Sessions objects.

CloseAll Method Example

This example closes all open sessions.

```
Sub Main()
   Dim Sys As Object
   Set Sys = CreateObject("EXTRA.System")
' Assumes one or more open sessions
   Sys.Sessions.CloseAll
End Sub
```

Col Property

Applies To Objects

Screen

Description

Returns or sets the column position of the cursor. Read-only for VT sessions.

Syntax

object.**Col**

Element	Description
object	The Screen object.

Col Property Example

This example displays the row and column cursor position of the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session

    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    CurrentRow = MyScreen.Row
    CurrentCol = MyScreen.Col

    MsgBox "The cursor is at " + Str$(CurrentRow) + "," _
    + Str$(CurrentCol) + "."
End Sub
```

ColorScheme Property

Applies To Objects

Session

Description

Returns the name of the color scheme, or sets the color scheme to be assigned to the session. Read-write.

Syntax

object.**ColorScheme**

Element	Description
object	The Session object.

Comments

Until the ColorScheme property has been set, it will return a null string. When set, the ColorScheme property affects the session in two ways:

- The name value of the ColorScheme property is stored temporarily; that is, the value is stored only as long as the session remains open. When you close the session and reopen it, the ColorScheme property will return a null string.
- The colors assigned to the session are stored permanently; that is, the colors are stored with the session until you reset the ColorScheme property. When you close the session and reopen it, the colors last assigned to the session are used.

ColorScheme Property Example

This example displays the name of the color scheme for the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
' Assumes the color scheme has previously been set
    MsgBox "The current color scheme is " + _
       Sess.ColorScheme + "."
End Sub
```

Cols Property

Applies To Objects

Screen

Description

Returns the number of columns in the presentation space. Read-only.

Syntax

object.**Cols**

Element	Description
object	The Screen object.

Cols Property Example

This example displays the number of columns in the Screen object.

```
Sub Main()
   Dim Sys As Object, Sess As Object, MyScreen As Object, _
   MyArea As Object
   Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
   Set Sess = Sys.ActiveSession
   Set MyScreen = Sess.Screen
   MsgBox "The screen has " + Str$(MyScreen.Cols) + " columns."
End Sub
```

Connected Property

Applies To Objects

Session

Description

Returns the connection status of the session -- TRUE if connected, FALSE if disconnected. The property also connects or disconnects the session. Read-write.

Syntax

object.**Connected**

Element	Description
object	The Session object.

Comments

Setting the Connected property to FALSE is equivalent to choosing the Disconnect command from the Session menu; setting it to TRUE is equivalent to choosing the Connect command.

Connected Property Example

This example identifies and displays the open sessions that are currently connected.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes one or more open sessions
    SessionCount = Sys.Sessions.Count
    For i = 1 to SessionCount
      If Sys.Sessions.Item(i).Connected Then
       Connected$ = Connected$ + Sys.Sessions.Item(i).Name + " "
      End If
    Next
    MsgBox "The following sessions are connected: " + Connected$
End Sub
```

ConnectionStatus Property

Applies To Objects

OIA

Description

Returns an integer indicating the status of the connection. Read-only.

Syntax

object.**ConnectionStatus**

Element	Description
object	The OIA object.

Comments

The following values indicate the connection state. To return the property, you can use either a constant or a value.

OIA Symbol	Constant	Value	Description
	xNO_STATUS	0	
4B	xAPP_OWNED	1	Session is connected to an application
4B	xSSCP	2	Control program owned
4B?	xUNOWNED	3	Session is not connected to an application

ConnectionStatus Property Example

This example uses the Screen object's OIA property to reference the OIA object. The OIA object is then used to return the session connection status (ConnectionStatus property).

```
Sub Main
    Dim Sys As Object
    Dim Sess As Object
    Dim MyScreen As Object
' This gets the System object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
' The following If statement displays status messages pertaining to
' your host connection.
    If MyScreen.OIA.ConnectionStatus = 1 Then
    MsgBox "Session currently connected to a mainframe application."
    ElseIf MyScreen.OIA.ConnectionStatus = 2 Then
        MsgBox "The control program has established contact."
    ElseIf MyScreen.OIA.ConnectionStatus = 3 Then
        MsgBox "Session not connected to a mainframe application."
    End If
End Sub
```

Copy Method

Applies To Objects

Area, Screen

Description

Copies the current selection to the Clipboard.

Syntax

object.**Copy**

Element	Description
object	The Area or Screen object.

Comments

To copy data from an Area or Screen object, you must first select the object with the Select method. (For the Screen object, you can also use the SelectAll method.)

Copy Method Example

This example first copies the entire session screen to the Clipboard, then copies an area of the screen to the Clipboard.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    My Area As Object

    ' Invoke the clipboard viewer
    Shell("clipbrd.exe")

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    ' This example demonstrates the Copy method for Screen objects.
    Set MyScreen = Sess.Screen
    MyScreen.SelectAll
    MyScreen.Copy

    MsgBox "The screen was copied to the clipboard. _
    Press to copy a smaller area ..."
    ' This example demonstrates the Copy method for Area objects.
    Set MyArea = MyScreen.Area(1,1,10,10 , ,)
    MyArea.Select
    MyArea.Copy
End Sub
```

Count Property

Applies To Objects

QuickPads, Sessions, Toolbars

Description

Returns the number of items in the collection of objects. Read-only.

Syntax

object.**Count**

Element	Description
object	The above-listed collection objects.

Comments

As described below, the meaning of the returned value depends on the object.

Object	Returned Value
Sessions	The number of open sessions.
QuickPads	The number of QuickPads available to the session.
Toolbars	The number of Toolbars available to the session.

Count Property Example

This example first counts and displays the names of the open sessions. It then counts and displays the names of the QuickPads visible for the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object, QPad As Object

    Set Sys = CreateObject("EXTRA.System")
'Assumes at least one open session

    ' This tests the Count property for a Sessions object.
    NumberOfSessions = Sys.Sessions.Count
    For i = 1 to NumberOfSessions
      OpenSessions$ = OpenSessions$ + _
      Sys.Sessions.Item(i).Name + " "
    Next

    MsgBox "The following sessions are open: " + OpenSessions$

    ' This tests the Count property for a QuickPads object.  This
    ' example works equally well for toolbars, replacing the
    ' QuickPads collection object with a Toolbars collection object.
    Set Sess = Sys.ActiveSession

    NumberOfQPads = Sess.QuickPads.Count
    For i = 1 to NumberOfQPads
      If Sess.QuickPads.Item(i).Visible Then
        VisiblePads$ = VisiblePads$ + _
        Sess.QuickPads.Item(i).Name + " "
      End If
    Next
    MsgBox "The following QuickPads are visible: " + VisiblePads$
End Sub
```

Cut Method

Applies To Objects

Area, Screen

Description

Cuts the current selection to the Clipboard. This method is not supported by VT sessions.

Syntax

object.**Cut**

Element	Description
object	The Area or Screen object.

Comments

To cut data from an Area or Screen object, you must first select the object with the Select method. (For the Screen object, you can also use the SelectAll method.)

Cut Method Example

This example first cuts an area of the screen to the Clipboard, then cuts the entire screen to the Clipboard.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    ' Invoke the clipboard viewer
    Shell("clipbrd.exe")

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This example demonstrates the Cut method for Area objects.
    Set MyArea = MyScreen.Area(1,1,10,10 , ,)
    MyArea.Select
    MyArea.Cut
    MsgBox "Press to Cut the entire Screen."

    ' This example demonstrates the Cut method for Screen objects.
    MyScreen.SelectAll
    MyScreen.Cut
End Sub
```

DefaultFilePath Property

Applies To Objects

System

Description

Sets or returns the default path specification as a string. This is the path used for opening session files (the default installation path is E!PC\SESSIONS). Read-write.

Syntax

object.**DefaultFilePath**

Element	Description
object	The System object.

DefaultFilePath Property Example

This example displays the default file path.

```
Sub Main()
    Dim Sys As Object

    Set Sys = CreateObject("EXTRA.System")
    DefaultPath$ = Sys.DefaultFilePath
    MsgBox "The current default file path is " + _
       DefaultPath$ + "."
End Sub
```

Delete Method

Applies To Objects

Area, Screen

Description

Deletes the current selection. This method is not supported by VT sessions.

Syntax

object.**Delete**

Element	Description
object	The Area or Screen object.

Comments

To delete data from an Area or Screen object, you must first select the object with the Select method. (For the Screen object, you can also use the SelectAll method.)

Delete Method Example

This example first deletes an area of the screen (from row 1, column 1 to row 5 column 5), then deletes an additional ten rows and columns.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This example demonstrates the Delete method for Area objects.
    Set MyArea = MyScreen.Area(1,1,5,5 , ,3)
    MyArea.Select
    MyArea.Delete
    MsgBox "Press to Delete a larger section of the screen."

    ' This example demonstrates the Delete method for Screen
    ' objects.
    MyScreen.Select10,10,20,20
    MyScreen.Delete
End Sub
```

EditScheme Property

Applies To Objects

Session

Description

Returns the name of the edit scheme, or sets the edit scheme to be assigned to the session. Read-write.

Syntax

object.**EditScheme**

Element	Description
object	The Session object.

Comments

Until the EditScheme property has been set, it will return a null string. When set, the EditScheme property affects the session in two ways:

- The name value of the EditScheme property is stored temporarily; that is, the value is stored only as long as the session remains open. When you close the session and reopen it, the EditScheme property will return a null string.
- The edit settings assigned to the session are stored permanently; that is, the settings are stored with the session until you reset the EditScheme property. When you close the session and reopen it, the edit settings last assigned to the session are used.

EditScheme Property Example

This example displays the name of the edit scheme for the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
' Assumes the edit scheme has previously been set
    MsgBox "The current edit scheme is " + Sess.EditScheme + "."
End Sub
```

ErrorStatus Property

Applies To Objects

OIA

Description

Returns an integer indicating any error condition. Read-only.

Syntax

object.**ErrorStatus**

Element	Description
object	The OIA object.

Comments

The following values indicate the errors returned. To return the property, you can use either a constant or a value.

OIA Symbol	Constant	Value	Description
	xNO_STATUS	0	
XPROG756	xPROG_CHECK	1	Configuration mismatch occurred
-+Z_510	xCOMM_CHECK	2	Communications hardware problem occurred
x0211	xMACHINE_CHECK	3	A problem with the physical connection occurred

ErrorStatus Property Example

This example uses the Screen object's OIA property to reference the OIA object. The OIA object is then used to return session errors (ErrorStatus property).

```
Sub Main
    Dim Sys As Object
    Dim Sess As Object
    Dim MyScreen As Object
' This gets the System object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
' The following If statement displays status messages pertaining to
' host/client communication.
    If MyScreen.OIA.ErrorStatus = 1 Then
        MsgBox "A configuration mismatch has occurred."
    ElseIf MyScreen.OIA.ErrorStatus = 2 Then
        MsgBox "A communications hardware problem has occurred."
    ElseIf MyScreen.OIA.ErrorStatus = 3 Then
        MsgBox "A physical connection problem has occurred."
    End If
End Sub
```

FileTransferHostOS Property

Applies To Objects

Session

Description

Returns or sets the host operating system used by file transfers executed through OLE Automation. Read-write.

Syntax

object.**FileTransferHostOS**

Element	Description
object	The Session object.

Comments

This property should be set prior to executing a file transfer. Valid values are as follows:

0 - CMS

1 - TSO

2 - CICS

Use this property in conjunction with the FileTransferScheme property.

FileTransferHostOS Property Example

```
Sub Main
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Select Case Sess.FileTransferHostOS
       Case 0
       MsgBox "The current file transfer HostOS is CMS"
       Case 1
       MsgBox "The current file transfer HostOS is TSO"
       Case 2
       MsgBox "The current file transfer HostOS is CICS"
    End Select
End Sub
```

FileTransferScheme Property

Applies To Objects

Session

Description

Returns the name of the file transfer scheme, or sets file transfer scheme to be assigned to the session. Read-write.

Syntax

object.**FileTransferScheme**

Element	Description
object	The Session object.

Comments

Until the FileTransferScheme property has been set, it will return a null string. When set, the FileTransferScheme property affects the session in two ways:

- The name value of the FileTransferScheme property is stored temporarily; that is, the value is stored only as long as the session remains open. When you close the session and reopen it, the FileTransferScheme property will return a null string.
- The file transfer settings assigned to the session are stored permanently; that is, the settings are stored with the session until you reset the FileTransferScheme property. When you close the session and reopen it, the file transfer settings last assigned to the session are used.

FileTransferScheme Property Example

This example displays the name of the file transfer scheme for the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
' Assumes the file transfer scheme has previously been set
    MsgBox "The current file transfer scheme is " _
    + Sess.FileTransferScheme + "."
End Sub
```

FullName Property

Applies To Objects

QuickPad, Session, System, Toolbar

Description

Returns a string specifying the path and filename. Read-only.

Syntax

object.**FullName**

Element	Description
object	One of the above-listed objects.

Comments

Depending on the object, the FullName property returns different values.

Object	Return Value
QuickPad	The name of the QuickPad. The default installation path is E!PC\SCHEMES; the default extension is EQP.
Session	Session filename. Session files are located in the E!PC\Sessions folder by default. Display session files have an EDP extension; printer session files have an EPP extension.
System	Program filename. The filename of the *EXTRA!* application is EXTRA.EXE, located in the folder E!PC.
Toolbar	The name of the Toobar. The default installation path is E!PC\SCHEMES; the default extension is ETB.

FullName Property Example

This example displays the path and filenames of the following objects: System, Session, QuickPad, and Toolbar.

```
Sub Main()
    Dim Sys As Object, Sess As Object, QPad As Object, _
    TBar As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session

    ' This example demonstrates how to use the FullName property
    ' with the System object.
    fqProgram$ = Sys.FullName
    MsgBox "The current fully qualified path and filename to E!PC _
    is " + fqProgram$ + "."

    ' This demonstrates how to use the FullName property with
    ' a Session object. Here, the ActiveSession property was used to
    ' get a session object.
    Set Sess = Sys.ActiveSession
    fqProfile$ = Sess.FullName
    MsgBox "The fully qualified path and filename to the current _
    session profile is " + fqProfile$ + "."

    ' This example demonstrates how to use the FullName property
    ' with a QuickPad object. Here, the ActiveSession property was
    ' used to get a session object.
    Set QPad = Sess.QuickPads.Item(1)
    QPad.Visible = True
    fqQPad$ = QPad.FullName
    MsgBox "The fully qualified path and filename to the current _
    QuickPad is " + fqQPad$ + "."
```

```
    ' This demonstrates how to use the FullName property with
    ' a QuickPad object. The ActiveSession property was used to
    ' get a session object.
    Set TBar = Sess.Toolbars.Item(1)
    TBar.Visible = True
    fqTBar$ = TBar.FullName
    MsgBox "The fully qualified path and filename to the current _
    Toolbar is " + fqTBar$ + "."

    MsgBox "Done."
End Sub
```

GetString Method

Applies To Objects

Screen

Description

Returns the text from the specified screen location.

Syntax

```
rc = object.GetString (Row, Col, Length, [Page])
```

Element	Description
rc	The returned screen text.
object	The Screen object.
Row	The row where the text string begins.
Col	The column where the text string begins.
Len	The length of the text string.
Page	VT session only—the screen page. **Note:** This parameter is ignored for release 6.0 of *EXTRA!*.

GetString Method Example

This example gets and displays the first half of the text on the first row of the screen.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This example demonstrates using the GetString
    ' method to capture the first half of the first line on a
    ' screen.
    row = 2
    col = 1
    halfLine = Int(MyScreen.Cols/2)
    MyString$ = MyScreen.GetString(row, col, halfLine)

    MsgBox "The first " + Str$(halfLine) + " characters of line " _
    + Str$(row) + " is [" + MyString$ + "] ..."
End Sub
```

Height Property

Applies To Objects

Session

Description

Returns or sets the height of the session window in pixels. Read-write.

Syntax

object.**Height**

Element	Description
object	The Session object.

Height Property Example

After returning the height of the active session window, this example reduces the height by 50%.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    StartingHeight = Sess.Height
    MsgBox "This will shrink the current window by 50% _
    (currenty " + Str$(StartingHeight) + " pixels)."
    Sess.Height = Int(StartingHeight/2)
End Sub
```

HideAll Method

Applies To Objects

QuickPads, Toolbars

Description

Hides all visible QuickPad or Toolbar objects.

Syntax

object.**HideAll**

Element	Description
object	The QuickPads or Toolbars object.

HideAll Method Example

This example displays a prompt for hiding all QuickPads and Toobars of the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    HideEm = MsgBox("Do you wish to hide all Toolbars and _
    QuickPads?", 1,"Hide All ...")
    If HideEm=1 Then 'The above message box returns 1 if Okay
                     'is chosen.
      Sess.QuickPads.HideAll
      Sess.Toolbars.HideAll
    End If
End Sub
```

HotSpotScheme Property

Applies To Objects

Session

Description

Returns the name of the HotSpot scheme, or sets the hotspot scheme assigned to the session. Read-write.

Syntax

object.**HotSpotScheme**

Element	Description
object	The Session object.

Comments

Until the HotSpotScheme property has been set, it will return a null string. When set, the HotSpotScheme property affects the session in two ways:

- The name value of the HotSpotScheme property is stored temporarily; that is, the value is stored only as long as the session remains open. When you close the session and reopen it, the HotSpotScheme property will return a null string.
- The hotspots assigned to the session are stored permanently; that is, the hotspots are stored with the session until you reset the HotSpotScheme property. When you close the session and reopen it, the hotspots last assigned to the session are used.

HotSpotScheme Property Example

This example displays the name of the hotspot scheme of the the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
    'Assumes an open session
    MsgBox "The current HotSpot scheme is " + _
    Sys.ActiveSession.HotSpotScheme + "."
End Sub
```

Item Method

Applies To Objects

Sessions, QuickPads, Toolbars

Description

Returns an element in the collection.

Syntax

object.**Item(***i***)**
-or-
object.**Item(***name***)**

Object	Description
object	One of the above-listed objects.
i	Numeric index of element in the collection.
name	Name of element in the collection.

Comments

The Item method is the default method for collection objects. The syntax of the following statements are equivalent:

object.Item(*name*)
-or-
object(*name*)

In the first statement, the item is called explicitly; in the second statement, the item is called implicitly.

For the Sessions collection, the numeric index refers to the sequence that the sessions were opened. For example, `Item(1)` refers to the first session opened, `Item(2)` refers to the second session opened and so on. For QuickPads and Toolbars, the numeric index refers to the order in which these objects were created.

You can assign the returned object to a variable with the Set statement.

Item Method Example

This example first displays the names of all open sessions, then displays the names of all visible QuickPads for the active session. It then goes on to show the difference between the explicit and implicit use of the Item method.

```
Sub Main()
    Dim Sys As Object, Sess As Object, QPad As Object

    Set Sys = CreateObject("EXTRA.System")
'Assumes at least one open session

    ' This tests the Item method for a Sessions object.
    NumberOfSessions = Sys.Sessions.Count
    For i = 1 to NumberOfSessions
      OpenSessions$ = OpenSessions$ + _
      Sys.Sessions.Item(i).Name + " "
    Next
    MsgBox "The following sessions are open: " + OpenSessions$

    ' This tests the Item method for a QuickPads object.  This
    ' example works equally well for toolbars. Substitute the
    ' QuickPads collection object with a Toolbars collection object.
    Set Sess = Sys.ActiveSession
    NumberOfQPads = Sess.QuickPads.Count
    For i = 1 to NumberOfQPads
      If Sess.QuickPads.Item(i).Visible Then
        VisiblePads$ = VisiblePads$ + Sess.QuickPads.Item(i).Name _
        + " "
      End If
    Next
    MsgBox "The following QuickPads are visible: " + VisiblePads$
```

```
    ' There are two ways to make an item call, implicitly or
    ' explicitly, and two ways to index it, a numeric index
    ' and a string index. The above examples show an explicit
    ' call with a numeric index.

    ' This example shows an implicit call with a numeric index.
    For i = 1 To NumberOfQPads
      If Sess.QuickPads(i).Visible Then
       VisiblePads$ = VisiblePads$ + Sess.QuickPads(i).Name + " "
      End If
    Next
    MsgBox "The following QuickPads are visible: " + VisiblePads$

    ' This example shows an explicit call with a string index.
    If Sess.QuickPads.Item("Aid").Visible Then
      MsgBox "The Aid Quick Pad is visible."
    End If

    ' This example shows an implicit call with a string index.
    If Sess.QuickPads("Aid").Visible Then
      MsgBox "The Aid Quick Pad is visible."
    End If
End Sub
```

JumpNext Method

Applies To Objects

Sessions

Description

Returns the next open session and gives it the focus.

Syntax

object.**JumpNext**

Element	Description
object	The Sessions object.

Comments

The JumpNext method switches focus in the order that the sessions are opened. If there is only one session open, this method is ignored.

You can assign the returned Session object to a variable with the Set statement.

JumpNext Method Example

This example gives focus to the next open session.

```
Sub Main()
   Dim Sys As Object, Sess As Object
   Set Sys = CreateObject("EXTRA.System")
' Assumes two or more open sessions
   Sys.Sessions.JumpNext
End Sub
```

KeyboardLocked Property

Applies To Objects

Session

Description

Returns or sets the keyboard input state for the session -- TRUE to block keyboard input; FALSE to accept keyboard input. Read-write.

Syntax

object.**KeyboardLocked**

Element	Description
object	The Session object.

Comments

By default, keyboard input from the user is accepted.

KeyboardLocked Property Example

This example tests the keyboard state of the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    If Sess.KeyboardLocked Then
      MsgBox "The keyboard is locked."
    Else
      MsgBox "The keyboard is not locked."
    End If
End Sub
```

KeyMap Property

Applies To Objects

Session

Description

Returns the name of the keyboard map, or sets the keyboard map to be assigned to the session. Read-write.

Syntax

object.**KeyMap**

Element	Description
object	The Session object.

Comments

The KeyMap property requires a valid keyboard map filename, including the EKM extension (for example, IBM 3191.EKM).

KeyMap Property Example

This example displays the name of the keyboard map for the active session.

```
Sub Main()
   Dim Sys As Object, Sess As Object
   Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
   Set Sess = Sys.ActiveSession
   MsgBox "The current keyboard map is " + Sess.KeyMap + "."
End Sub
```

Left Property

Applies To Objects

Area, Session

Description

For the Area object, the Left property returns or sets the screen column where the area begins. For the Session object, the Left property returns or sets the horizontal position of the session, in pixels. Read-write.

Syntax

object.**Left**

Element	Description
object	The Area or Session object.

Comments

Depending on the object, the Left property has different meanings.

Object	Meaning
Area	The column where the area starts, expressed as an integer.
Session	The number of pixels between the left edge of the session window and the left edge of the screen.

For the Area object, you can set the Left property to shrink or expand the object. For example, if you initially defined an area with a left column of two, you can set the left column to four, thereby shrinking the size of the area. You can also adjust the size of the Area object with the Bottom, Top, and Right properties.

For the Session object, you can set the Left property to change a session window's horizontal position on the screen. To change a session window's vertical position, set the Top property.

Left Property Example

This example first resets the horizontal position of the active session, then resets the column where the area begins.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This moves the session to the left of the screen.
    Sess.Left = 1

    Set MyArea = MyScreen.Area(1,1,MyScreen.Rows,1,,3)
    MsgBox "The text in column 1 is: " + MyArea.Value

    MyArea.Left = 2
    MyArea.Right = 2

    MsgBox "The test in column 2 is: " + MyArea.Value
End Sub
```

MoveRelative Method

Applies To Objects

Screen

Description

Moves the cursor a specified number of rows and columns from its current position. This method is not supported by VT sessions.

Syntax

```
object.MoveRelative NumOfRows, NumOfCols
```

Element	Description
object	The Screen object.
NumrOfRows	The number of rows to move.
NumOfCols	The number of columns to move.

Comments

To move the cursor to an absolute row and column, use the MoveTo method.

MoveRelative Method Example

This example moves the cursor to twenty-five random positions.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This causes the cursor to run all over the screen
    For i = 1 To 25
        Randomize
        dX = Int(Rnd() * 10) - 5
        dY = Int(Rnd() * 10) - 5
        MyScreen.MoveRelative dX, dY
    Next
End Sub
```

MoveTo Method

Applies To Objects

Screen

Description

Moves the cursor to the specified location. This method is not supported by VT sessions.

Syntax

object.**MoveTo** *Row*, *Col*

Element	Description
object	The Screen object.
Row	The row location.
Col	The column location.

Comments

To move the cursor to a position relative to its current one, use the MoveRelative method.

MoveTo Method Example

The example moves the cursor to a random row and column on the screen.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    For i = 1 to 10
      Randomize
      NewRow = Int(MyScreen.Rows*Rnd())+1
      NewCol = Int(MyScreen.Cols*Rnd())+1
      MsgBox "Move to " + Str$(NewRow) + "," + _
         Str$(NewCol) + "."
      MyScreen.MoveTo NewRow,NewCol
    Next
End Sub
```

Name Property

Applies To Objects

QuickPad, Screen, Session, System, Toolbar

Description

Returns the name of the object as a string. Read-only. See Comments for details.

Syntax

object.**Name**

Element	Description
object	Any of the above-listed objects.

Comments

As shown below, the Name property has different meanings for different objects.

Object	Meaning
QuickPad	The name of the QuickPad.
Screen	The name of the screen recorded from the session window. (From the Tools menu, choose Record Pages.) The name appears in the status line of the session window. If the screen does not have a name, the property returns an empty string. You can move to a named screen with the NavigateTo method.
Session	The name of the session.
System	The name of the *EXTRA!* program.
Toolbar	The name of the Toolbar.

Name Property Example

This example demonstrates the use of the Name property with System, Session, QuickPad, and Toolbar objects.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    ' This gets the System object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This is an example of how to use the Name property for the
    ' System object.
    SysName$ = Sys.Name
    MsgBox "The Sys Name is " + SysName$ + "."

    ' This is an example of how to use the Name property for the
    ' Screen object.
    ScreenName$ = MyScreen.Name
    MsgBox "The Screen Name is " + ScreenName$ + "."

    ' This is an example of how to use the Name property for a
    ' Session object.
    SessionName$ = Sys.ActiveSession.Name
    MsgBox "The Session Name is " + SessionName$ + "."

       ' This is an example of how to use the Name property for a
       ' QuickPad object. This also shows how the QuickPads object
       ' (a collection object)might be used.
```

```
    For i = 1 to Sess.QuickPads.Count
     If Sess.QuickPads.Item(i).Visible then
       MsgBox "The QuickPad " + Sess.QuickPads.Item(i).Name + _
       " is visible."
     else
       MsgBox "The QuickPad " + Sess.QuickPads.Item(i).Name + " _
       is NOT visible."
     end if
    Next

    ' This is an example of how to use the Name property for
    ' a Toolbar object. This also shows how the Toolbars object
    ' (a collection object) might be used.
    For i = 1 to Sess.ToolBars.Count
     If Sess.ToolBars.Item(i).Visible then
       MsgBox "The Toolbar " + Sess.ToolBars.Item(i).Name +  _
       " is visible."
     else
       MsgBox "The Toolbar " + Sess.ToolBars.Item(i).Name +  _
       " is NOT visible."
     end if
    Next

    MsgBox "Done."
End Sub
```

NavigateTo Method

Applies To Objects

Session

Description

Navigates to a specified host screen, recorded from a session window.

Syntax

```
rc = object.NavigateTo ([screenName])
-or-
object.NavigateTo [screenName]
```

Element	Description
rc	The return value.
object	The Session object.
screenName	The name of the screen to navigate to.

Return Value	Description
TRUE	Successful
FALSE	The specified screen cannot be located. -or- User canceled out of the dialog box.

Comments

To use this method, you must first record pages from a session.

From the Tools menu, choose Record Pages.

If you do not specify *screenName*, a dialog box appears for selecting a screen.

NavigateTo Method Example

The example navigates to a host screen named Page002.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    ' Note that Page002 must have already been recorded
    ' for this to work.
    Sess.NavigateTo ("Page002")
End Sub
```

OIA Object

Description

The OIA Object provides information about the Operator Information Area (OIA) of the host screen. The host OIA provides mainframe and mainframe connection status information.

Properties

ConnectionStatus	Value
ErrorStatus	XStatus
Updated	

OIA Property

Applies To Objects

Screen

Description

Returns an OIA object.

Syntax

object.**OIA**

Element	Description
object	The Screen object.

OIA Property Example

This example uses the Screen object's OIA property to reference the OIA object. The OIA object is then used to:

- return the OIA image as a character string for parsing and evaluation (Value property),
- determine whether or not the OIA has been updated since the last time the OIA object was accessed (Updated property),
- return the status of the XCLOCK portion of the OIA (XStatus property),
- return the session connection status (ConnectionStatus property),
- and, return session errors (ErrorStatus property).

```
Sub Main
    Dim Sys As Object
    Dim Sess As Object
    Dim MyScreen As Object
' This gets the System object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
' This returns (and displays) the current OIA message as a string.
    OIAValue$ = MyScreen.OIA.Value
    MsgBox "Current OIA message codes for this session: " + _
        OIAValue$
' The following If statement displays a message if the
' OIA has been updated.
    If MyScreen.OIA.Updated = 1 Then
        MsgBox "The OIA has been updated!"
    End If
' The following If statement displays status messages pertaining to
' user-entered host commands and/or data.
    If MyScreen.OIA.XStatus = 1 Then
```

```
      MsgBox "You have entered an invalid number."
   ElseIf MyScreen.OIA.XStatus = 2 Then
   MsgBox "You have entered non-numeric data in a numeric field."
   ElseIf MyScreen.OIA.XStatus = 3 Then
   MsgBox "You have attempted to enter data in a protected field."
   ElseIf MyScreen.OIA.XStatus = 4 Then
   MsgBox "You have attempted to type past the end of a field."
   ElseIf MyScreen.OIA.XStatus = 5 Then
   MsgBox "The host is busy processing your request."
   ElseIf MyScreen.OIA.XStatus = 6 Then
   MsgBox "The function you requested is unavailable."
   ElseIf MyScreen.OIA.XStatus = 7 Then
   MsgBox "Unable to print to requested printer."
   ElseIf MyScreen.OIA.XStatus = 8 Then
   MsgBox "The system has locked your keyboard during processing."
   ElseIf MyScreen.OIA.XStatus = 9 Then
      MsgBox "You have entered an invalid character."
   End If
' The following If statement displays status messages pertaining to
' your host connection.
   If MyScreen.OIA.ConnectionStatus = 1 Then
   MsgBox "Session connected to mainframe application."
   ElseIf MyScreen.OIA.ConnectionStatus = 2 Then
      MsgBox "The control program has established contact."
   ElseIf MyScreen.OIA.ConnectionStatus = 3 Then
      MsgBox "Session not connected to mainframe application."
   End If
' The following If statement displays status messages pertaining to
' host/client communication.
```

```
    If MyScreen.OIA.ErrorStatus = 1 Then
       MsgBox "A configuration mismatch has occurred."
    ElseIf MyScreen.OIA.ErrorStatus = 2 Then
       MsgBox "A communications hardware problem has occurred."
    ElseIf MyScreen.OIA.ErrorStatus = 3 Then
       MsgBox "A physical connection problem has occurred."
    End If
End Sub
```

Open Method

Applies To Objects

Sessions

Description

Returns an existing session and adds it to the Sessions collection.

Syntax

object.**Open** [*fileName*]

Element	Description
object	The Sessions object.
fileName	The filename of the session, which can include the path. The file extension is optional.

Comments

You can assign the returned Session object to a variable with the Set statement.

Open Method Example

The example opens a session called Session1.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
    Sys.Sessions.Open("Session1.EDP")
End Sub
```

PageRecognitionTime Property

Applies To Objects

Session

Description

Returns or sets the recognition time used by the Record Pages feature. PageRecognitionTime can be an integer between 1 and 20 and indicates the number of seconds you want to give the host to settle and "recognize" the host page during the record and playback procedures. Read-write.

Syntax

object.**PageRecognitionTime**

Element	Description
object	The Session object.

PageRecognitionTime Property Example

After determining the current PageRecognitionTime value, this example prompts the user to set a new PageRecognitionTime.

```
Sub Main()
    Dim Sys As Object
    Dim Sess As Object
    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
    ' These lines set up the strings to be used in the
    ' InputBox dialog
    InputPrompt$ = "The current page recognition time value is " _
        + Sess.PageRecognitionTime
    InputPrompt$ = InputPrompt$ + "  Enter a value to change it."
    Title$ = "Set page recognition time value"
    Default$ = Str$(Sess.PageRecognitionTime)
    NewTime$ = InputBox$( InputPrompt$, Title$, Default$)
    ' The PageRecognitionTime property is used to set the host page
    ' recognition time value.
    Sess.PageRecognitionTime = Val(NewTime$)
    MsgBox "The new value is " + Sess.PageRecognitionTime + "."
End Sub
```

Parent Property

Applies To Objects

Area, QuickPad, QuickPads, Screen, Session, Sessions, System, Toolbar, Toolbars

Description

Returns the parent of the specified object. Read-only.

Syntax

object.**Parent**

Element	Description
object	One of the above-listed objects.

Comments

The following table lists the parent of each object.

Object	Parent
System	System
Sessions	System
Session	Sessions
Screen	Session
Area	Screen
QuickPads	Session
QuickPad	QuickPads
Toolbars	Session
Toolbar	Toolbars

Parent Property Example

For each of the *EXTRA!* objects, this example shows how to return its parent object.

```
Sub Main()
    Dim Sys As Object, AllSess As Object, Sess As Object
    Dim MyScreen As Object, MyArea As Object
    Dim TBars As Object, QPads As Object, MyTBar As Object, _
    MyQPad As Object

    Set Sys = CreateObject("EXTRA.System")
    Set AllSess = Sys.Sessions
' Assumes an open session. Retrieve objects.
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(1, 1, 10, 10, , 3)
    Set TBars = Sess.Toolbars
    Set QPads = Sess.QuickPads
    Set MyTBar = TBars(1)
    Set MyQPad = QPads(1)

    ' The System object is its own parent.
    MsgBox "System.Parent = " + Sys.Parent.Name

    'The parent of a Sessions object is also the System object.
    MsgBox "Sessions.Parent = " + AllSess.Parent.Name

    ' The parent of a Session object is a Sessions object,
    ' whose parent is ...
    MsgBox Sess.Parent.Parent.Name

    ' The parent of a Screen object is a Session object,
    ' whose parent is ...
    MsgBox MyScreen.Parent.Parent.Parent.Name
```

```
    ' The parent of a Toolbars object or QuickPads object is also a
    ' Session, whose parent is ...
    MsgBox TBars.Parent.Parent.Parent.Name
    MsgBox QPads.Parent.Parent.Parent.Name

    ' The parent of an Area object is a Screen object,
    ' whose parent is ...
    MsgBox MyArea.Parent.Parent.Parent.Parent.Name

    ' The parent of a Toolbar object is a Toolbars object,
    ' whose parent is ...
    MsgBox MyTBar.Parent.Parent.Parent.Parent.Name
    ' The parent of a QuickPad object is a QuickPads object,
    ' whose parent is ...
    MsgBox MyQPad.Parent.Parent.Parent.Parent.Name
End Sub
```

Paste Method

Applies To Objects

Area, Screen

Description

For the Area object, this method pastes Clipboard text into the object. For the Screen object, this method pastes Clipboard text at the current position or over the current selection.

Syntax

object.**Paste**

Element	Description
object	The Area or Screen object.

Comments

You must first copy or cut data to the Clipboard, then select the Area or Screen object (with the Select method) to which you want to paste.

Paste Method Example

This example selects a host from the SSCP screen of a DEMO3270.EDP session.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
    ' Assumes an open session
    Set Sess = Sys.ActiveSession

    ' This example is meant to be run from a DEMO3270 session.
    ' This example demonstrates the Paste method for Screen objects.
    ' Set MyScreen = Sess.Screen
    MyScreen.Select 7,10,7,10
    MyScreen.Copy
    MyScreen.Select 23,6,23,6
    MyScreen.Paste

    ' This example demonstrates the Paste method for Area objects.
    Set MyArea = MyScreen.Area(9,10,9,10,,3)
    MyArea.Select
    MyArea.Copy
    Set MyArea = MyScreen.Area(23,6,23,6,,3)
    MyArea.Select
    MyArea.Paste
End Sub
```

Path Property

Applies To Objects

Session, System

Description

For the System object, this property returns the path of the *EXTRA!* executable file. For the Session object, this property returns the path of the session file. The filename is not included. Read-only.

Syntax

object.**Path**

Element	Description
object	The System or Session object.

Comments

Session files are located in the Sessions directory by default. A session is referenced by the name of the file that stores its settings. Display sessions have an EDP extension; printer sessions have an EPP extension.

Path Property Example

This example first displays the path of the System object, then the active Session object.

```
Sub Main()
    Dim Sys As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session

    ' This example demonstrates how to use the Path property with
    ' the System object.
    CurrentProgramPath$ = Sys.Path
    MsgBox "The current path to EXTRA! is " + CurrentProgramPath$ _
    + "."

    ' This example demonstrates how to use the Path property with a
    ' Session object. Here, the ActiveSession property was used to
    ' get a session object.
    CurrentProfilePath$ = Sys.ActiveSession.Path
    MsgBox "The path to the current session profile _
    is " + CurrentProfilePath$ + "."

    MsgBox "Done."
End Sub
```

PutString Method

Applies To Objects

Screen

Description

Puts text in the specified location on the screen. This method is not supported by VT sessions.

Syntax

object.**PutString** *String*,[*Row*],[*Col*]

Element	Description
object	The Screen object.
String	Text that you want to put on the screen.
Row	The row in which to put the text. Optional. If this parameter is not supplied, the current cursor row position is used.
Col	The column in which to put the text. Optional. If this parameter is not supplied, the current cursor column position is used.

PutString Method Example

This example puts the string "Hello" on every row of the screen.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    For i = 1 to MyScreen.Rows
      Hello$ = " Hello "
      MyScreen.PutString Hello$,i,1
    Next
End Sub
```

QuickPad Object

Description

Provides access to a specific QuickPad.

Properties

Application

FullName

Name

Parent

Visible

Comments

Using the Item method of the QuickPads collection object, you can return a specific QuickPad object.

QuickPads Object

A collection object consisting of individual QuickPad objects.

Properties	Methods
Application	HideAll
Count	Item
Parent	

Comments

You can retrieve a QuickPads object with the QuickPads property of the Session object. For example, as shown in the following code, the QuickPads object is returned and assigned to the object variable QPcoll.

```
Dim ses As Object, QPcoll As Object
Set ses = GetObject("Sess1.Ses")
Set QPcoll = ses.QuickPads
```

QuickPads Property

Applies To Objects

Session

Description

Returns the QuickPads collection containing the individual QuickPad objects that are currently available to a session. Read-only.

Syntax

```
Set rc = object.QuickPads
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Session object.

Comments

This collection consists of all of the QuickPads that are in the local and remote paths. If no QuickPads are available, the object is still returned, but its Count property is 0.

Once the QuickPads collection object is returned, you can access a specific Quickpad object. You can do this by using the Item method of the QuickPads collection object. For example, the following compound statement returns a reference to a QuickPad object.

```
Set StandardQP = Extra.QuickPads.Items(1)
```

If the returned QuickPad is not visible, set its Visible property to TRUE.

QuickPads Property Example

This example determines how many QuickPads are available to the active session (with the aid of the Count property), and then lists them (with the aid of the Item method).

```
Sub Main()
    Dim Sys As Object, Sess As Object, QPads As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

' Assumes that there are one or more QuickPads available
    QPadCount = Sess.QuickPads.Count

    For i = 1 to QPadCount
      QPadNames$ = QPadNames$ + Sess.QuickPads.Item(i).Name + " "
    Next

    MsgBox "The number of QuickPads = " + QPadCount + ". _
    They are: " + QPadNames$
End Sub
```

Quit Method

Applies To Objects

System

Description

Closes all sessions and *EXTRA!* programs.

Syntax

object.**Quit**

Element	Description
object	The System object.

Comments

This method is equivalent to the CloseAll method used with the Sessions object.

Quit Method Example

This example closes all *EXTRA!* applications and sessions.

```
Sub Main()
    Dim Sys As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes one or more open session
    Sys.Quit
End Sub
```

ReceiveFile Method

Applies To Objects

Session

Description

Receives a file from the host.

Syntax

```
rc = object.ReceiveFile ([PCFileName [,HostFileName]])
-or-
object.ReceiveFile [PCFileName [,HostFileName]]
```

Element	Description
rc	The return value.
object	The Session object.
PCFileName	The name of the file after it's transferred to the PC.
HostFileName	The name of the file to transfer from the host.

Return Value	Description
TRUE	Successful
FALSE	The specified file cannot be located. -or- User canceled out of Transfer dialog box. -or- The file transfer was unsuccessful for any reason.

Comments

If you do not specify *HostFileName* or *PCFileName*, the Transfer dialog box appears where you can select files.

Note that the SendFile and ReceiveFile methods do not support FT5250 (SQL) file transfers. 3270 IND$FILE transfers and FTP file transfers (using any host) are supported, however.

ReceiveFile Method Example

The example transfers the file “test text” from the host and renames it “test2.txt” on the PC. Based on the return value of ReceiveFile, the message box indicates success or failure of the transfer.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    Sess.FileTransferScheme = "Text Default"
    Sess.FileTransferHostOS = 0       '0 = CMS

    Recv = Sess.ReceiveFile("c:\test2.txt","test text")
    If Recv Then MsgBox ("Okay") Else MsgBox ("Error")
End Sub
```

Right Property

Applies To Objects

Area

Description

Returns or sets the column, specified as an integer, where the Area ends. Read-write.

Syntax

object.**Right**

Element	Description
object	The Area object.

Comments

You can set the Right property to shrink or expand an Area object. For example, if you initially defined an area with a right column of ten, you can set the right column to fifteen, thereby expanding the size of the area.

You can also adjust the size of the Area object with the Left, Top, and Bottom properties.

Right Property Example

This example uses the Left and Right properties to reset the column where the area begins.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(1,1,MyScreen.Rows,1,,3)

    MsgBox "The text in column 1 is: " + MyArea.Value

    ' This moves the session to the left of the screen.
    MyArea.Left = 2
    MyArea.Right = 2

    MsgBox "The test in column 2 is: " + MyArea.Value
End Sub
```

Row Property

Applies To Objects

Screen

Description

Returns or sets the row position of the cursor. Read-only for VT sessions.

Syntax

object.**Row**

Element	Description
object	The Screen object.

Row Property Example

This example displays the row and column cursor position of the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session

    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    CurrentRow = MyScreen.Row
    CurrentCol = MyScreen.Col

    MsgBox "The cursor is at " + Str$(CurrentRow) + "," + _
    Str$(CurrentCol) + "."
End Sub
```

Rows Property

Applies To Objects

Screen

Description

Returns the number of rows in the presentation space. Read-only.

Syntax

object.**Rows**

Element	Description
object	The Screen object.

Rows Property Example

This example displays the number of rows in the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    MsgBox "The screen has " + Str$(MyScreen.Rows) + " rows."
End Sub
```

Save Method

Applies To Objects

Session

Description

Saves the current settings of the session.

Syntax

object.**Save**

Element	Description
object	The Session object.

Comments

Use the Saved property to determine the save state of the session.

Save Method Example

This example saves the settings of the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    Sess.Save
End Sub
```

SaveAs Method

Applies To Objects

Session

Description

Saves a copy of the specified session to a new file.

Syntax

object.**SaveAs** [*FileName*]

Element	Description
object	The Session object.
FileName	The name of the file where the session is saved.

Comments

If you do not specify *FileName*, the SaveAs dialog box prompts for one.

SaveAs Method Example

This example saves a copy of the active session to a new session named SessionX.

```
Sub Main()
    Dim Sys As Object, Sess As Object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    Sess.SaveAs ("SessionX")
End Sub
```

Saved Property

Applies To Objects

Session

Description

Returns the save status of the session -- TRUE if the session has been saved since it was last modified, FALSE if not. Read-only.

Syntax

object.**Saved**

Element	Description
object	The Session object.

Saved Property Example

This example illustrates the two values of the Saved property, TRUE or FALSE.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    If Sess.Saved Then
      MsgBox "This session has not been changed since it was _
      last saved."
    Else
      MsgBox "This session HAS been changed since it was _
      last saved."
    End If
End Sub
```

Screen Object

Description

Provides access to the contents of the host screen's presentation space.

Properties	Methods	
Application	Area	Select
Col	Copy	SelectAll
Cols	Cut	SendInput
Name	Delete	SendKeys
Row	GetString	WaitForCursor
OIA	MoveRelative	WaitForCursorMove
Parent	MoveTo	WaitForKeys
Rows	Paste	WaitForStream
Selection	PutString	WaitForString
Updated	Search	WaitHostQuiet

Comments

You can retrieve a Screen object with the Screen property of the Session object. For example, as shown in the following code, the Screen object is returned and assigned to the object variable SessionScreen.

```
Dim ses As Object, SessionScreen As Object
Set ses = GetObject("Sess1.Ses")
Set SessionScreen = ses.Screen
```

To access the presentation space with a Screen object, you must know the exact number of rows and columns that an emulated terminal provides. For example, if a session emulates a terminal supporting 24 rows by 80 columns, you can reference presentation space positions from row 1, column 1 to row 24, column 80. For VT sessions, you can also specify the page.

Screen Property

Applies To Objects

Session

Description

Returns the Screen object associated with the session.

Syntax

```
Set rc = object.Screen
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Session object.

Screen Property Example

This example moves a selected line from the top of the screen to the bottom.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object
    Dim MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
    ' Assumes an open session
    Set Sess = Sys.ActiveSession

    ' This example moves a selected line from the top of the screen
    ' to the bottom.
    Set MyScreen = Sess.Screen
    For i = 1 to MyScreen.Rows
      MyScreen.Select i,1,i,MyScreen.Cols
    Next
End Sub
```

Search Method

Applies To Objects

Screen

Description

Returns an Area object with the text specified in the search.

Syntax

```
Set rc = object.Search(Text[,Row][,Col][,Page])
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Screen object.
Row	The row where the search begins.
Col	The column where the search begins.
Page	VT session only—the screen page where the search begins. **Note:** This parameter is ignored for release 6.0 of *EXTRA!*.

Comments

If the optional parameters are used, Screen is searched from the specified starting position. Otherwise, the entire Screen object is searched.

If Search finds the specified text, the coordinate properties (Left, Top, Right, Bottom) of the returned Area object are set to the starting and ending row and column positions of the text. The Value property of the Area object is set to the text located at those coordinates. If the Screen changes at those coordinates, the Value property of the Area changes.

If Search does not find the specified text, the Area object's Value property is set to an empty string, its Type property is set to xNONE, and its coordinate properties are set to -1.

Search Method Example

This example first finds the input location on the host demonstration screen (using the SlideShow connection type). It then move the cursor to the screen position where data is entered.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This will find the input location on the first
    ' Slide Show screen.
    MyScreen.MoveTo 1, 1
    Set MyArea = MyScreen.Search("==>")
    MyScreen.MoveTo MyArea.Bottom, MyArea.Right + 2
End Sub
```

Select Method

Applies To Objects

Area, Screen

Description

For the Area object, the method selects the object. For the Screen object, the method selects the area defined by the coordinates and returns an Area object.

Area Syntax

```
object.Select
```

Element	Description
object	The Area object.

Screen Syntax

```
Set rc = object.Select(StartRow, _
       StartCol,EndRow,EndCol[,Page])
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Screen object.
StartRow	The row where the selection begins.
StartCol	The column where the selection begins.
EndRow	The row where the selection ends.

Element (cont.)	Description (cont.)
EndCol	The column where the selection ends.
Page	VT session only—the screen page. **Note:** This parameter is ignored for release 6.0 of *EXTRA!*.

Select Method Example

The example first shows how to select an Area object, then a Screen object.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    Set MyArea = MyScreen.Area(5, 5, 10, 10, , 3)
    MyArea.Select
    MsgBox "Select the Area object ..."

    Set MyArea = MyScreen.Select(11, 11, 20, 20)
    MsgBox "Select the Screen object ..."
End Sub
```

SelectAll Method

Applies To Objects

Screen

Description

Selects the entire screen and returns an Area object.

Syntax

Set *rc* = *object*.**SelectAll**

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Screen object.

SelectAll Method Example

This example uses SelectAll to create an Area object with the same coordinates as the Screen object.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
    ' This selects the entire screen.
    Set MyArea = MyScreen.SelectAll
End Sub
```

Selection Property

Applies To Objects

Screen

Description

Returns an Area object representing the area of the screen currently selected by the user. Read-only.

Syntax

```
Set rc = object.Selection
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Screen object.

Selection Property Example

Returning an Area object from a user's selection, the example displays the coordinates of the object, referred to as MyArea.

```
Sub Main()
   Dim Sys As Object, Sess As Object, MyScreen As Object, MyArea As
Object

   Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
   Set Sess = Sys.ActiveSession
   Set MyScreen = Sess.Screen

   ' This will return the coordinates of the area selected in the
   ' current session. Note, that if nothing is selected the area
   ' will be empty, i.e., its coordinates will all be -1.
   Set MyArea = MyScreen.Selection
   MyString$ = MyString$ + "Left = " + Str$(MyArea.Left)
   MyString$ = MyString$ + "; Right = " + Str$(MyArea.Right)
   MyString$ = MyString$ + "; Top = " + Str$(MyArea.Top)
   MyString$ = MyString$ + "; Bottom = " + Str$(MyArea.Bottom)
   MsgBox MyString$
End Sub
```

SendFile Method

Applies To Objects

Session

Description

Sends a file from the PC to the host.

Syntax

```
rc = object.SendFile ([PCFileName [,HostFileName]])
-or-
object.SendFile [PCFileName [,HostFileName]]
```

Element	Description
rc	The return value.
object	The Session object.
PCFileName	The name of the file to send to the host.
HostFileName	The name of the file after it's transferred to the host.

Return Value	Description
TRUE	Successful
FALSE	The specified file cannot be located. -or- User canceled out of Transfer dialog box. -or- The file transfer was unsuccessful for any reason.

Comments

If you do not specify *HostFileName* or *PCFileName*, the Transfer dialog box prompts for files.

Note that the SendFile and ReceiveFile methods do not support FT5250 (SQL) file transfers. 3270 IND$FILE transfers and FTP file transfers (using any host) are supported, however.

SendFile Method Example

The example transfers the file "test2.txt" from the PC and renames it "test text" on the host. Based on the return value of SendFile, the message box indicates success or failure of the transfer.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    Sess.FileTransferScheme = "Text Default"
    Sess.FileTransferHostOS = 0       '0 = CMS

    Sent = Sess.SendFile("c:\test.txt","test text")
    If Sent Then MsgBox ("Sent Okay.") _
    Else MsgBox ("Error while sending.")
End Sub
```

SendInput Method

Applies To Objects

Screen

Description

Sends the specified text to the Screen object, simulating incoming data from the host.

Note: This method is for VT sessions only. (Use the Session.Type property to determine the session type.)

Syntax

object.**SendInput** *Text*

Element	Description
object	The Screen object.
Text	String of ANSII characters, including control characters like carriage returns and line feeds.

SendInput Method Example

This example inputs a string to a VT terminal session.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
' Assumes an open session
    Set MyScreen = Sess.Screen

    ' This will put Hello User! in the upper left of a VT terminal
    VT$ = Chr$(27) + "[H"
    MyScreen.SendInput VT$ + " Hello User! "
End Sub
```

SendKeys Method

Applies To Objects

Screen

Description

Sends keystrokes to the host, including function keys. The keystrokes appear to the session as if they were manually entered by a user.

Syntax

object.**SendKeys(***String***)**

Element	Description
object	The Screen object.
String	String of keystrokes, up to 255. You can specify host function keys as well as alphanumeric keys.

Comments

To pause your application while the host processes the transmitted keystrokes, use a wait method, such as WaitForCursor.

Host function keys are specified with mnemonics, which are different for each type of host -- 3270, 5250, or VT. Use the applicable table below to identify the mnemonic that represents a particular 3270, 5250, or VT function key. When you specify a mnemonic, enclose it in angle brackets (< >). For example, in the following statement, the Screen object uses the SendKeys method to transmit a Command 1 function to the host.

```
Screen.SendKeys <Pf1>
```

3270 and 5250 Function Keys Table

3270 Function Key	Mnemonic	5250 FunctionKey	Mnemonic
Attention	Attn	Attention	Attn
Backspace	BackSpace	Backspace	BackSpace
Backtab (Left Tab)	BackTab	Backtab (Left Tab)	BackTab
Caps Lock	CapsLock	Begin Line	BeginLine
Clear	Clear	Clear	Clear
Cursor Down	Down	Command 1	Pf1
Cursor Left	Left	Command 2	Pf2
Cursor Left 2 Columns	Left2	Command 3	Pf3
Cursor Right	Right	Command 4	Pf4
Cursor Right 2 Columns	Right2	Command 5	Pf5
Cursor Select	CursorSelect	Command 6	Pf6
Cursor Up	Up	Command 7	Pf7
Delete Char	Delete	Command 8	Pf8
Duplicate	Dup	Command 9	Pf9
Enter	Enter	Command 10	Pf10
Erase EOF	EraseEOF	Command 11	Pf11
Erase Input	EraseInput	Command 12	Pf12
FieldMark	FieldMark	Command 13	Pf13
Home	Home	Command 14	Pf14
Insert	Insert	Command 15	Pf15
Left Tab (Back Tab)	BackTab	Command 16	Pf16
New Line	NewLine	Command 17	Pf17
Pen Select	PenSel	Command 18	Pf18
Print	Print	Command 19	Pf19
Reset	Reset	Command 20	Pf20
Shift	ShiftOn	Command 21	Pf21
System Request	SysReq	Command 22	Pf22
Tab (Right Tab)	Tab	Command 23	Pf23
Pa1	Pa1	Command 24	Pf24
Pa2	Pa2	Cursor Down	Down
Pa3	Pa3	Cursor Down 2 rows	Down2

3270 and 5250 Function Keys Table (cont.)

3270 Function Key	Mnemonic	5250 FunctionKey	Mnemonic
Pf1	Pf1	Cursor Left	Left
Pf2	Pf2	Cursor Left 2 columns	Left2
Pf3	Pf3	Cursor Right	Right
Pf4	Pf4	Cursor Right 2 columns	Right2
Pf5	Pf5	Cursor Select	CursorSelect
Pf6	Pf6	Cursor Up	Up
Pf7	Pf7	Cursor Up 2 rows	Up2
Pf8	Pf8	Delete Char	Delete
Pf9	Pf9	Duplicate	Dup
Pf10	Pf10	End of Line	EndLine
Pf11	Pf11	Enter	Enter
Pf12	Pf12	Erase EOF	EraseEOF
Pf13	Pf13	Erase EOL	EraseEOL
Pf14	Pf14	Erase Input	EraseInput
Pf15	Pf15	Field Exit	FieldExit
Pf16	Pf16	Field Mark	FieldMark
Pf17	Pf17	Field Minus	FieldMinus
Pf18	Pf18	Field Plus	FieldPlus
Pf19	Pf19	Help	Help
Pf20	Pf20	Home	Home
Pf21	Pf21	Insert Mode	InsertMode
Pf22	Pf22	Insert Toggle	Insert

3270 and 5250 Function Keys Table (cont.)

3270 Function Key	Mnemonic	5250 FunctionKey	Mnemonic
Pf23	Pf23	Left Tab (Back Tab)	BackTab
Pf24	Pf24	New Line	NewLine
		Pa1	@x
		Pa2	@y
		Pa3	@z
		Print	Print
		Reset	Reset
		Right Tab (Tab)	Tab
		Roll Down	RollDown
		Roll Up	RollUp
		System Request	SysReq
		Tab (Right Tab)	Tab
		Test Request	TestRequest

VT Function Keys Table

Note: The keypad functions will send either numbers or control sequences, depending on the current mode of the keypad.

VT Function Keys	Mnemonic	VT Function Keys	Mnemonic
Backspace	Backspace	Find	Find
Break Signal	Break	Insert	Insert Here
Compose Sequence	Compose	Hold	Hold
Cursor Down	Down	Keypad -	Keypad -
Cursor Left	Left	Keypad ,	Keypad ,
Cursor Right	Right	Keypad Enter	Keypad Enter
Cursor Up	Up	Keypad .	Keypad .
Escape	Esc	Keypad 0	Keypad 0
F1	F1	Keypad 1	Keypad 1
F2	F2	Keypad 2	Keypad 2
F3	F3	Keypad 3	Keypad 3
F4	F4	Keypad 4	Keypad 4
F5	F5	Keypad 5	Keypad 5
F6	F6	Keypad 6	Keypad 6
F7	F7	Keypad 7	Keypad 7
F8	F8	Keypad 8	Keypad 8
F9	F9	Keypad 9	Keypad 9
F10	F10	Next Screen	Next
F11	F11	PF1	PF1
F12	F12	PF2	PF2
F13	F13	PF3	PF3
F14	F14	PF4	PF4
F15	F15	Previous Screen	Prev
F16	F16	Print Screen	Print
F17	F17	Remove	Remove
F18	F18	Select	Select
F19	F19	Setup	Setup
F20	F20	Tab	Tab

SendKeys Method Example

The example uses SendKeys to page through three host demonstration screens (using the SlideShow connection type). Note that with the exception of the first instance of SendKeys, this method is only executed if the WaitForCursor method returns TRUE.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    Sys.TimeoutValue = 9000

    ' This will page through a couple of "slide show" host screens.
    MyScreen.SendKeys ("a<Enter>")
    Found1st = MyScreen.WaitForCursor(20, 16)
    If Found1st Then
      MyScreen.SendKeys ("user1<Enter>")
      Found2nd = MyScreen.WaitForCursor(23, 1)
      If Found2nd Then
        MyScreen.SendKeys ("logoff<Enter>")
        Found3rd = MyScreen.WaitForCursor(23, 6)
      End If
    End If
    If Not (Found1st And Found2nd And Found3rd) _
    Then MsgBox "Error navigating host screens."
End Sub
```

Session Object

Description

Provides access to host data and *EXTRA!* functionality. The Session object contains sub-objects, such as the Screen object and the Toolbar object.

Properties

Application	KeyboardLocked	Screen
ColorScheme	KeyMap	Toolbars
Connected	Left	Top
EditScheme	Name	Type
FileTransferHostOS	PageRecognitionTime	Visible
FileTransferScheme	Parent	Width
FullName	Path	WindowState
Height	QuickPads	
HotSpotScheme	Saved	

Methods

Activate	Save
Close	SaveAs
NavigateTo	SendFile
ReceiveFile	

Comments

The most direct way to retrieve a Session object is with the GetObject function, which retrieves a reference to an object from a file. You can also use the Sessions object's Open method.

Sessions Object

Description

A collection object consisting of sessions.

Properties	Methods
Application	CloseAll
Count	Item
Parent	JumpNext
	Open

Comments

The Sessions collection reflects individual sessions that are currently open. By accessing the Sessions collection, you can cycle through the sessions and perform actions.

Sessions Property

Applies To Objects

System

Description

Returns the Sessions collection containing the individual Session objects that are currently open. Read-only.

Syntax

```
Set rc = object.Sessions
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The System object.

Comments

If no sessions are opened, the object is still returned, but its Count property is 0.

Sessions Property Example

This example determines how many sessions are open (with the aid of the Count property), and then lists them (with the aid of the Item method).

```
Sub Main()
    Dim Sys As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes one or more open sessions

    ' This example uses the Sessions property to:
    ' 1) get the number of sessions (with the aid of the Count
    ' property),
    ' 2) list them (with the aid of the Item method).

    SessionCount = Sys.Sessions.Count

    For i = 1 to SessionCount
      SessNames$ = SessNames$ + Sys.Sessions.Item(i).Name + " "
    Next

    MsgBox "The number of sessions = " + SessionCount + ". _
    They are: " + SessNames$
End Sub
```

System Object

Description

Top-level object, providing access to all objects in *EXTRA!*.

Properties		Methods
ActiveSession	Sessions	Quit
Application	TimeoutValue	ViewStatus
DefaultFilePath	Version	
FullName		
Name		
Parent		
Path		

TimeoutValue Property

Applies To Objects

System

Description

Sets or returns the number of milliseconds used by Wait operations (for example, `Screen.WaitForString` and `Area.WaitUntilChanged`). Read-write.

Syntax

object.**TimeoutValue**

Element	Description
object	The System object.

Comments

The initial default timeout value is 30,000 milliseconds (30 seconds). If you change TimeoutValue, the new value becomes the default.

TimeoutValue Property Example

After determining the current timeout value, this example prompts the user to set a new timeout.

```
Sub Main()
    Dim Sys As Object

    Set Sys = CreateObject("EXTRA.System")

    ' These lines set up the strings to be used in the
    ' InputBox dialog
    InputPrompt$ = "The current timeout value is " + _
    Sys.TimeoutValue
    InputPrompt$ = InputPrompt$ + "  Enter a value to change it."
    Title$ = "Set Timeout Value"
    Default$ = Str$(Sys.TimeoutValue)

    NewTimeout$ = InputBox$( InputPrompt$, Title$, Default$)

    ' The TimeoutValue property is used to set the timeout value.
    Sys.TimeoutValue = Val(NewTimeout$)
    MsgBox "The new value is " + Sys.TimeoutValue + "."
End Sub
```

Toolbar Object

Description

Provides access to a specific Toolbar.

Properties

Application

FullName

Name

Parent

Visible

Comments

Using the Item method of the Toolbars collection object, you can return a specific Toolbar object.

Toolbars Object

Description

A collection object consisting of individual Toolbar objects.

Properties	Methods
Application	HideAll
Count	Item
Parent	

Comments

You can retrieve a Toolbars object with the Toolbars property of the Session object. For example, as shown in the following code, the Toolbars object is returned and assigned to the object variable TBcoll.

```
Dim ses As Object, TBcoll As Object
Set ses = GetObject("Sess1.Ses")
Set TBcoll = ses.Toolbars
```

Toolbars Property

Session

Description

Returns the Toolbars collection containing the individual Toolbar objects that are currently available to the session. Read-only.

Syntax

```
Set rc = object.Toolbars
```

Element	Description
Set	The Set statement, required for assigning an object reference to a variable.
rc	The object variable for referencing the returned object.
object	The Session object.

Comments

This collection consists of all of the Toolbars that are in the local and remote paths. If no Toolbars are available, the object is still returned, but its Count property is 0.

Once the Toolbars collection object is returned, you can access a specific Toolbar object. You can do this by using the Item method of the Toolbars collection object. For example, the following compound statement returns a reference to a Toolbar object.

```
Set StandardTB = Extra.Toolbars.Items(1)
```

If the returned Toolbar is not visible, set its Visible property to TRUE.

Toolbars Property Example

This example determines how many Toolbars are available to the active session (with the aid of the Count property), and then lists them (with the aid of the Item method).

```
Sub Main()
    Dim Sys As Object, Sess As Object, TBars As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    ' This example uses the Toolbars property to:
    ' 1) get the number of Toolbars (with the aid of the Count
    '    property)
    ' 2) list them (with the aid of the Item method).

    TBarCount = Sess.Toolbars.Count

    For i = 1 to TBarCount
      TBarNames$ = TBarNames$ + Sess.Toolbars.Item(i).Name + " "
    Next

    MsgBox "The number of Toolbars = " + TBarCount + ". _
    They are: " + TBarNames$
End Sub
```

Top Property

Applies To Objects

Area, Session

Description

For the Area object, the Top property returns or sets the row where the area begins. For the Session object, the Top property returns or sets the vertical position of the session, in pixels. Read-write.

Syntax

object.**Top**

Element	Description
object	The Area or Session object.

Comments

Depending on the object, the Top property has different meanings.

Object	Meaning
Area	The row where the area starts, expressed as an integer.
Session	The number of pixels between the top edge of the session window and the top of the screen.

For the Area object, you can set the Top property to shrink or expand the object. For example, if you initially defined an area with a top row of two, you can set the top row to four, thereby shrinking the size of the area. You can also adjust the size of the Area object with the Bottom, Left, and Right properties.

For the Session object, you can set the Top property to change a session window's vertical position on the screen. To change a session window's horizontal position, set the Left property.

Top Property Example

This example uses Top and Bottom properties with an Area object to narrow a selection on the screen. Then the example uses the Top property with a Session object to move the session around on the screen.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
    ' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This illustrates the Top and Bottom properties for an Area
    ' object. For demonstration purposes, the Select method is used,
    ' but it is not required for setting Top and Bottom properties.
    MsgBox "Press to demonstrate the Top and Bottom properties for _
    Area objects."
    Set MyArea = MyScreen.Area(1, 1, MyScreen.Rows, _
    MyScreen.Cols,,3)
    MyArea.Select
    For i = 1 to Int(MyScreen.Rows/2)
      MyArea.Top = MyArea.Top + 1
      MyArea.Select
      MyArea.Bottom = MyArea.Bottom - 1
      MyArea.Select
    Next

    ' This demonstrates the Top property for a Session object.
    MsgBox "Press to demonstrate the Top property for _
    Session objects."
```

```
    Sess.Top = 50
    MsgBox "The session top is now at 50.  Press to move session _
    top to 1"
    Sess.Top = 1
    MsgBox "The session top is now at 1.  Press to move session _
    top to 100"
    Sess.Top = 100
End Sub
```

Type Property

Applies To Objects

Area, Session

Description

For the Area object, the Type property determines how the Area coordinates (top, left, bottom, right) are interpreted when the object is selected. Read-write.

For the Session object, the Type property returns a value indicating the session type -- 3270, 5250, or VT. Read-only.

Syntax

object.**Type**

Element	Description
object	The Area or Session object.

Comments

The following values indicate the Area type. To set the property, you can use either a constant or a value.

Constant	Value	Area Type
xBLOCK	3	The Area is selected as a rectangular range of characters. There is no line wrapping.
xSTREAM	2	The Area is selected as a continuous stream of characters, from the top left coordinate to the bottom right coordinate. If the Area consists of more than one line, then the selection wraps to the right of each line
xPOINT	1	The Area is selected as a single point. The bottom and right coordinates are ignored. Not supported by VT session.

(cont.)

Constant	Value	Area Type
xNONE	0	Invalidates selection of the Area. The xNONE value is returned for an Area object with a Value property set to an empty string. Such an object will be created by the Screen object's Search method—if the search string is not found. You cannot change the Value property if set to an empty string.

The following values indicate the Session type. (These values have constant equivalents.)

Constant	Value	Session Type
3270SESSION	1	3270
5250SESSION	2	5250
VTSESSION	3	VT

Type Property Example

This example shows how the Type property can be used with the Session object and the Area object.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    'Using Type with Session object
    Select Case Sess.Type
    Case 1
       MsgBox "This is a 3270 session."
    Case 2
       MsgBox "This is a 5250 session."
    Case 3
       MsgBox "This is a VT session."
    End Select

    'Using Type with Area object
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(5, 5, 10, 10, , 3)
```

```
    Select Case MyArea.Type
    Case 0
       MsgBox "The Area type is xNONE."
    Case 1
       MsgBox "The Area type is xPOINT."
    Case 2
       MsgBox "The Area type is xSTREAM."
    Case 3
       MsgBox "The Area type is xBLOCK."
    End Select
End Sub
```

Updated Property

Applies To Objects

Screen, OIA

Description

Returns TRUE if the Screen or OIA object has been updated since the last time this property was checked. If the object has not been updated, the property returns FALSE. Read-only.

Syntax

object.**Updated**

Element	Description
object	The Screen or OIA object.

Updated Property Example

This example illustrates the two values of the Updated property, TRUE or FALSE.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' This macro is meant to be run from the start of an IBM SSCP
    ' screen. If the' enter key does not force a screen update,
    ' soon enough, the second Updated check will still return false.
    If MyScreen.Updated Then
      MsgBox "The screen has been updated."
    Else
      MsgBox "The screen has NOT been updated."
    End If

    MyScreen.SendKeys ("a<Enter>")

    If MyScreen.Updated Then
      MsgBox "The screen has been updated."
    Else
      MsgBox "The screen has NOT been updated."
    End If

End Sub
```

Value Property

Applies To Objects

Area, OIA

Description

Returns (for both Area and OIA objects) or sets (for the Area object only) the text in the Area or Operator Information Area (OIA). Value is read-write for the Area object and read-only for the OIA object.

Syntax

object.**Value**

Element	Description
object	The Area or OIA object.

Comments

Value is the default property of the Area and OIA objects.

Value Property Example

This example uses the Value property to log on to a simulated host (using the SlideShow connection type). Note that the While...Wend statement is used to test if the logon screen has appeared. Alternatively, you can use the WaitForString method to test this condition.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object, _
    MyArea As Object

    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
' Assumes an open session
    Set MyScreen = Sess.Screen
    Set MyArea = MyScreen.Area(23, 6, 23, 6, , 3)

    ' This macro is designed for the "slide show" session
    MyArea.Value = "a"
    MyScreen.SendKeys ("<Enter>")

    Set MyArea = MyScreen.Area(20, 2, 20, 7, , 3)
    nCounter = 0
    maxCounter = 500
    Wait$ = "USERID"
    While ((Watch$ <> Wait$) And (nCounter < maxCounter))
       Watch$ = MyArea.Value
       nCounter = nCounter + 1
    Wend
```

```
    If nCounter = maxCounter Then
      MsgBox Wait$ + " not found in time."
    Else
      Set MyArea = MyScreen.Area(20, 16, 20, 23, , 3)
      MyArea = "user1"
      ' This is identical to MyArea.Value = "user1"
      MyScreen.SendKeys ("<Enter>")
    End If
End Sub
```

Version Property

Applies To Objects

System

Description

Returns a string identifying the version of *EXTRA!*. Read-only.

Syntax

object.**Version**

Element	Description
object	The System object.

Version Property Example

This example displays the current version of *EXTRA!*.

```
Sub Main()
   Dim Sys As Object
   Set Sys = CreateObject("EXTRA.System")

   MsgBox "The current version of E!PC is " + Sys.Version + "."
End Sub
```

ViewStatus Method

Applies To Objects

System

Description

Starts the Status program.

Syntax

object.**ViewStatus**

Element	Description
object	The System object

Comments

The Status application cannot be closed programmatically; it must be closed manually.

ViewStatus Method Example

This example starts the Status application.

```
Sub Main()
    Dim Sys As Object
    Set Sys = CreateObject("EXTRA.System")

    Sys.ViewStatus
End Sub
```

Visible Property

Applies To Objects

QuickPad, Session, Toolbar

Description

Sets the object to visible or invisible, or returns its visibility status -- TRUE if visible, FALSE if invisible. Read-write.

Syntax

object.**Visible**

Element	Description
object	Any of the above-listed objects.

Comments

By default, a new session does not appear. To make it visible, set the property to TRUE. Likewise, to make a QuickPad or Toolbar appear, set its Visible property to TRUE.

Visible Property Example

Using the Visible property, this example indicates which QuickPads are visible and invisible, then indicates which sessions are visible and invisible.

```
Sub Main()
Dim Sys As Object, AllSess As Object, Sess As Object, _
QPads As Object

   Set Sys = CreateObject("EXTRA.System")
   Set AllSess = Sys.Sessions
' Assumes an open session
   Set Sess = Sys.ActiveSession
   Set QPads = Sess.QuickPads

   ' This example demonstrates the Visible property with QuickPad
   ' objects. This example works equally well for Toolbars by
   ' replacing the QuickPads object with a Toolbars object.
   For i = 1 To QPads.Count
    If QPads.Item(i).Visible Then
      VisQPads$ = VisQPads$ + QPads.Item(i).Name
    Else
      InvisQPads$ = InvisQPads$ + QPads.Item(i).Name
    End If
   Next
   MsgBox "The following QuickPads are visible: " + VisQPads$
   MsgBox "The following QuickPads are NOT visible: " + InvisQPads$
```

```
    ' Likewise, this example demonstrates the Visible property with
    ' Session objects.
    For i = 1 To AllSess.Count
     If AllSess.Item(i).Visible Then
       VisSess$ = VisSess$ + AllSess.Item(i).Name
     Else
       InvisSess$ = InvisSess$ + AllSess.Item(i).Name
     End If
    Next
    MsgBox "The following Sessions are visible: " + VisSess$
    MsgBox "The following Sessions are NOT visible: " + InvisSess$
End Sub
```

WaitForCursor Method

Applies To Objects

Screen

Description

Waits until the cursor is at the specified location. The method will wait for the amount of time set in System.TimeoutValue.

Syntax

```
rc = object.WaitForCursor (Row [,Col] [,Page]])
-or-
object.WaitForCursor Row [,Col] [,Page]]
```

Element	Description
rc	The return value.
object	The Screen object.
Row	The row where you want the cursor to appear.
Col	The column where you want the cursor to appear.
Page	VT session only—the screen page. **Note:** This parameter is ignored for release 6.0 of *EXTRA!*.

Return Value	Description
TRUE	Cursor is at the specified location within the time specified by System.TimeoutValue.
FALSE	Cursor is not at the specified location within the time specified by System.TimeoutValue.

Comments

You can set the number of milliseconds to wait by setting the System object's TimeoutValue property.

WaitForCursor Method Example

The example uses WaitForCursor and SendKeys to page through three host demonstration screens (using the SlideShow connection type). Note that with the exception of the first instance of SendKeys, this method is only executed if the WaitForCursor method returns TRUE.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    Sys.TimeoutValue = 9000

    ' This will page through a couple of "slide show" host screens.
    MyScreen.SendKeys ("a<Enter>")
    Found1st = MyScreen.WaitForCursor(20, 16)
    If Found1st Then
      MyScreen.SendKeys ("user1<Enter>")
      Found2nd = MyScreen.WaitForCursor(23, 1)
      If Found2nd Then
        MyScreen.SendKeys ("logoff<Enter>")
        Found3rd = MyScreen.WaitForCursor(23, 6)
      End If
    End If
    If Not (Found1st And Found2nd And Found3rd) _
    Then MsgBox "Error navigating host screens."
End Sub
```

WaitForCursorMove Method

Applies To Objects

Screen

Description

Waits until the cursor has moved the specified number of rows and columns from its current position. The method will wait for the amount of time set in System.TimeoutValue.

Syntax

```
rc = object.WaitForCursorMove (NumofRows [,NumofCols]
[,Pages]])
-or-
object.WaitForCursor NumofRows [,NumofCols]
[,NumofPages]]
```

Element	Description
rc	The return value.
object	The Screen object.
NumofRows	The number of rows to move.
NumofCols	The number of columns to move.
NumofPages	VT session only—the number of screen pages to move. **Note:** This parameter is ignored for release 6.0 of *EXTRA!*.

Return Value	Description
TRUE	Cursor has moved the specified amount within the time specified by System.TimeoutValue.
FALSE	Cursor has not moved the specified location within the time specified by System.TimeoutValue.

Comments

You can set the number of milliseconds to wait by setting the System object's TimeoutValue property.

WaitForCursorMove Method Example

This example moves the screen cursor and enters input to navigate through a couple of host demonstration screens (using the SlideShow connection type).

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
' Assumes an open session
    Set MyScreen = Sess.Screen
    Sys.TimeoutValue = 9000
    ' This will page through a couple of "slide show" host screens.
    MyScreen.SendKeys ("a<Enter>")
    ' This uses row and column parameters
    Move1st = MyScreen.WaitForCursorMove(-3, 10)
    If Move1st Then
      MyScreen.SendKeys ("user1<Enter>")
      ' This uses only the row optional parameter
      Move2nd = MyScreen.WaitForCursorMove(3)
      If Move2nd Then
       MyScreen.SendKeys ("logoff<Enter>")
       ' This also uses row parameter only
       Move3rd = MyScreen.WaitForCursorMove(0, 5)
      End If
    End If
    If Not (Move1st And Move2nd And Move3rd) _
    Then MsgBox "Error navigating host screens."
End Sub
```

WaitForKeys Method

Applies To Objects

Screen

Description

Waits until the user presses a key or for the specified time to elapse before the program will continue processing.

Syntax

```
rc = object.WaitForKeys ([Timeout [,UserKeys]])
-or-
object.WaitForKeys [Timeout [,UserKeys]]
```

Element	Description
rc	The return value.
object	The Screen object.
Timeout	The time for the Screen object to wait, in milliseconds. If no timeout value is specified, the Screen object waits until a key is pressed.
UserKeys	One or more keys that the user must press for the program to continue processing. If no string is specified, any keystroke will continue program processing.

Return Value	Description
UserKeys	The *UserKeys* keys that the user pressed.
empty string	Indicates that no key was pressed within the specified time or that the key that the user pressed was not one of the *UserKeys* you supplied in the WaitForKeys method syntax.

WaitForKeys Method Example

The example waits for the user to press the "a" character or for nine seconds to elapse.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    ' Method 1: As set in TimeoutValue property,
    ' WaitForKeys will wait for nine seconds until
    ' the "a" key is pressed
    Sys.TimeoutValue = 9000
    Wait4Keys$ = MyScreen.WaitForKeys("a")
    MsgBox "WaitForKeys returned " + Wait4Keys$ + "."

    ' Method 2: As set in the Timeout parameter,
    ' WaitForKeys will wait for nine seconds until
    ' the "a" key is pressed
    Wait4Keys$ = MyScreen.WaitForKeys(9000, "a")
    MsgBox "WaitForKeys returned " + Wait4Keys$ + "."
End Sub
```

WaitForStream Method

Applies To Objects

Screen

Description

Waits until the specified text appears in the host data stream. The method will wait for the amount of time set in System.TimeoutValue.

Syntax

```
rc = object.WaitForStream (Text [,IdleTime])
-or-
object.WaitForStream Text [,IdleTime]
```

Element	Description
rc	The return value.
object	The Screen object.
Text	The text string from the host.
IdleTime	The number of milliseconds when no host data is received after the specified text string has been located.

Return Value	Description
TRUE	Text is received in the data stream within the time specified by System.TimeoutValue.
FALSE	Text is not received in the data stream within the time specified by System.TimeoutValue.

Comments

If you specify idle time, the method will wait until until the following two conditions are met: first, the specified text string is received, and second, after the specified text is received, no additional text is received for the specified idle time.

WaitForStream Method Example

The example uses WaitforStream and SendKeys to page through host demonstration screens (using the SlideShow connection type). Note that with the exception of the first instance of SendKeys, this method is only executed if the WaitForStream method returns TRUE.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
' Assumes an open session
    Set MyScreen = Sess.Screen
' WaitForStream will wait for up to 9 seconds.
    Sys.TimeoutValue = 9000
    ' This will page through a couple of "slide show" host screens.
    MyScreen.SendKeys ("a<Enter>")
    ' WaitForStream uses the optional idle time parameter. It will
    ' wait until the string is received followed by 2 seconds of
    'idle time, or until the System timeout value is reached.
    Found1st = MyScreen.WaitForStream("COMMAND  ===>", 2000)
    If Found1st Then
      MyScreen.SendKeys ("user1<Enter>")
      Found2nd = MyScreen.WaitForStream("Ready;")
      If Found2nd Then
       MyScreen.SendKeys ("logoff<Enter>")
       Found3rd = MyScreen.WaitForStream("USSMSG10")
      End If
    End If
    If Not (Found1st And Found2nd And Found3rd) _
    Then MsgBox "Error navigating host screens."
End Sub
```

WaitForString Method

Applies To Objects

Screen

Description

Waits until the specified text appears on the screen. The Screen object will wait for the amount of time set in System.TimeoutValue.

Syntax

rc = *object*.**WaitForString (***Text* [,*Row* [,*Col* [,*Page*]]]**)**

-or-

object.***WaitForString*** *String* [,*Row* [,*Col* [,*Page*]]]

Element	Description
rc	The return value.
object	The Screen object.
String	The text string that you want the Screen object to wait for.
Row	The row where you expect the string to appear.
Col	The column where you expect the string to appear.
Page	VT session only—the screen page. **Note:** This parameter is ignored for release 6.0 of *EXTRA!*.

Return Value	Description
TRUE	The Screen object has received the text string in the specified location within the time specified by System.TimeoutValue.
FALSE	The Screen object has not received the text string in the specified location within the time specified by System.TimeoutValue.

Comments

If you don't specify a screen location (row, column, or page), WaitForString can receive the text string in any location to be successful.

WaitForString Method Example

The example uses WaitforString and SendKeys to page through host demonstration screens (using the SlideShow connection type). Note that with the exception of the first instance of SendKeys, this method is only executed if the WaitForString method returns TRUE.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
    Set Sess = Sys.ActiveSession
' Assumes an open session
    Set MyScreen = Sess.Screen

    Sys.TimeoutValue = 9000

    ' This will page through a couple of "slide show" host screens.
    MyScreen.SendKeys ("a<Enter>")
    ' This uses row and column optional parameters
    Found1st = MyScreen.WaitForString("COMMAND  ===>", 23, 2)
    If Found1st Then
      MyScreen.SendKeys ("user1<Enter>")
      ' This uses neither the row nor the column optional parameters
      Found2nd = MyScreen.WaitForString("Ready;")
      If Found2nd Then
        MyScreen.SendKeys ("logoff<Enter>")
        ' This uses only the row optional parameter
        Found3rd = MyScreen.WaitForString("USSMSG10", 2)
      End If
    End If
    If Not (Found1st And Found2nd And Found3rd) _
    Then MsgBox "Error navigating host screens."
End Sub
```

WaitHostQuiet Method

Applies To Objects

Screen

Description

Waits for the host to not send data for a specified number of milliseconds. The Screen object will wait for host compliance for the amount of time set in System.TimeoutValue.

Syntax

rc = *object*.**WaitHostQuiet (**[*SettleTime*]**)**
-or-
object.**WaitHostQuiet** [*SettleTime*]

Element	Description
rc	The return value.
object	The Screen object.
SettleTime	The amount of time, in milliseconds, that the host should remain "quiet." If *SettleTime* is not specified, a default time of 5000 (5 seconds) is assumed.

Return Value	Description
TRUE	The host was quiet for *SettleTime* milliseconds.
FALSE	The host was NOT quiet for *SettleTime* milliseconds.

Comments

This method can be used to allow time for the host to "settle down" during potentially busy periods. For instance, in a 3270 session, WaitHostQuiet is useful after issuing a SendKeys call that contains an Aid key. The Aid key can result in a change to the host screen and the host will issue Xclock (busy) messages during this time. Issuing a WaitHostQuiet call after the SendKeys call pauses processing of your macro until the host has remained quiet for 5 seconds.

There are two values that can affect the performance of WaitHostQuiet. These values will need to be set according to the performance of your particular host environment. The SettleTime parameter controls the amount of time you want the host to "be quiet." the default (5000) means "wait until the host has been quiet for 5 seconds." In some instances, you may want to pass in 0. For example, in a 3270 session, WaitHostQuiet(0) means "wait until the XClock clears."

The second value that can affect the performance of WaitHostQuiet is System.TimeoutValue. This value controls how long the macro should keep waiting for the quiet condition to occur before returning TRUE or FALSE. For example, if System.TimeoutValue was set to 30000 (30 seconds) and a WaitHostQuiet(10000) is issued, you are requesting to "wait for the host to be quiet for 10 seconds and if this hasn't happened in 30 seconds, return FALSE."

Tip: WaitForString, WaitForStream, WaitForCursor, and WaitForCursorMove are other possible options for determining when the host has finished processing.

WaitHostQuiet Method Example

The example uses WaitHostQuiet and SendKeys to page through host demonstration screens (using the SlideShow connection type). Note that with the exception of the first instance of SendKeys, this method is only executed if the WaitHostQuiet method returns TRUE.

```
Sub Main()
    Dim Sys As Object, Sess As Object, MyScreen As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen

    Sys.TimeoutValue = 9000

    ' This will page through a couple of "slide show" host screens.
    MyScreen.SendKeys ("a<Enter>")
    Wait1st = MyScreen.WaitHostQuiet(9000)
    If Wait1st Then
      MyScreen.SendKeys ("user1<Enter>")
      Wait2nd = MyScreen.WaitHostQuiet(9000)
      If Wait2nd Then
        MyScreen.SendKeys ("logoff<Enter>")
        Wait3rd = MyScreen.WaitHostQuiet(9000)
      End If
    End If
    If Not (Wait1st And Wait2nd And Wait3rd) _
    Then MsgBox "Error navigating host screens."
End Sub
```

Width Property

Applies To Objects

Session

Description

Returns or sets the width of the session window in pixels. Read-write.

Syntax

object.

Element	Description
object	The Session object.

Width Property Example

After returning the width of the active session window, this example reduces the width by 50%.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession

    StartingWidth = Sess.Width
    MsgBox "This will shrink the current window by 50% _
    (currenty " + StartingWidth + " pixels)."
    Sess.Width = Int(StartingWidth/2)
End Sub
```

WindowState Property

Applies To Objects

Session

Description

Returns or sets the state of the session window -- normal, maximized, or minimized. Read-write.

Syntax

object.**WindowState**

Element	Description
object	The Session object.

Comments

The following values indicate the window state. To set the property, you can use either a constant or a value.

Constant	Value	Window State
xMINIMIZED	0	Minimized
xNORMAL	1	Normal (not maximized or minimized)
xMAXIMIZED	2	Maximized

WindowState Property Example

This example sets the three possible window states for the active session.

```
Sub Main()
    Dim Sys As Object, Sess As Object

    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    MsgBox "Press for Minimized state ..."
    Sess.WindowState = 0
    MsgBox "Press for Normal state ..."
    Sess.WindowState = 1
    MsgBox "Press for Maximized state ..."
    Sess.WindowState = 2
End Sub
```

XStatus Property

Applies To Objects

OIA

Description

Returns an integer indicating the status of the XCLOCK. Read-only.

Syntax

object.**XStatus**

Element	Description
object	The OIA object.

Comments

The following values indicate the XCLOCK state. To return the property, you can use either a constant or a value.

OIA Symbol	Constant	Value	Description
	xNO_STATUS	0	
X ¥ #	xINVALID_NUM	1	User entered an invalid number
X ¥ NUM	xNUM_ONLY	2	User entered non-numeric data in a numeric field
X < ¥ >	xPROTECTED_FIELD	3	User typed in a protected field
X ¥ >	xPAST_EOF	4	User typed past the end of the field
X ( )	xBUSY	5	Host is busy
X - f	xINVALID_FUNC	6	Function is invalid
X ¥ X	xUNAUTHORIZED_PRINTER	7	Unauthorized printer requested
XSYSTEM	xSYSTEM	8	System locked during processing
X-s	xINVALID_CHAR	9	An invalid character was entered

XStatus Property Example

This example uses the Screen object's OIA property to reference the OIA object. The OIA object is then used to return the status of the XCLOCK portion of the OIA (XStatus property).

```
Sub Main
    Dim Sys As Object
    Dim Sess As Object
    Dim MyScreen As Object
' This gets the System object
    Set Sys = CreateObject("EXTRA.System")
' Assumes an open session
    Set Sess = Sys.ActiveSession
    Set MyScreen = Sess.Screen
' The following If statement displays status messages pertaining to
' user-entered host commands and/or data.
    If MyScreen.OIA.XStatus = 1 Then
       MsgBox "You have entered an invalid number."
    ElseIf MyScreen.OIA.XStatus = 2 Then
    MsgBox "You have entered non-numeric data in a numeric field."
    ElseIf MyScreen.OIA.XStatus = 3 Then
    MsgBox "You have attempted to enter data in a protected field."
    ElseIf MyScreen.OIA.XStatus = 4 Then
    MsgBox "You have attempted to type past the end of a field."
    ElseIf MyScreen.OIA.XStatus = 5 Then
       MsgBox "The host is busy processing your request."
    ElseIf MyScreen.OIA.XStatus = 6 Then
       MsgBox "The function you requested is unavailable."
    ElseIf MyScreen.OIA.XStatus = 7 Then
       MsgBox "Unable to print to requested printer."
```

```
    ElseIf MyScreen.OIA.XStatus = 8 Then
    MsgBox "The system has locked your keyboard during processing."
    ElseIf MyScreen.OIA.XStatus = 9 Then
       MsgBox "You have entered an invalid character."
    End If
End Sub
```

Glossary

Boolean Value: One of two possible values, TRUE or FALSE.

Collection: An object that manages a set of related objects. For example, the Sessions object manages individual Session objects; the Toolbars and QuickPad objects manage Toolbar and Quickpad objects. A collection object makes it easy to perform repetitive actions on all of the objects within the collection, such as closing all open sessions.

Color Scheme: A group of settings that specify the colors to be used for a display session. Only one color scheme can be assigned to a session at any one time.

Default Property or Method: A property or method of an object that does not have to be explicitly stated.

The Name property is the default for the Session object. The following two statements both return the name value for a Session object referred to as Ses1:

```
SesName = Ses1.Name
–or–
SesName = Ses1
```

The Item method is the default method of collection objects. The following two statements close the third session in the Sessions collection.

```
Sessions.Item(3).Close
–or–
Sessions(3).Close
```

Desktop Layouts: An *EXTRA!* Basic macro that opens one or more *EXTRA!* Personal Client windows in a prearranged position on your screen.

Edit Scheme: A group of settings that specify operational parameters of the following Edit menu commands {bmc emdash.bmp} Cut, Copy, Paste, and Clear. Only one edit scheme can be assigned to a display session at any one time.

***EXTRA!* BASIC Macro:** A sequence of commands saved in a file, with an extension of .EBM. The commands consist of statements and functions of the *EXTRA!* Basic language, included with *EXTRA!* Personal Client.

File Transfer Scheme: A group of settings that specify the file transfer parameters to be used for a display session. Only one file transfer scheme can be assigned to a session at any one time.

Hotspot Scheme: A group of settings that specify the Hotspots to be used for a display session. A Hotspot is an area of the screen defined to accept mouse input. When clicked, the Hotspot executes a host function. Only one HotSpot scheme can be assigned to a session at any one time.

Keyboard Map: A group of settings that specify the assignment of host characters and functions to PC keys.

Method: A command that directs an *EXTRA!* Personal Client object to perform an action. For example, the expression **Session.Activate** makes the specified session the active one.

Object Hierarchy: The hierarchical organization of *EXTRA!* Personal Client objects, which defines the access path to objects. The System object is at the highest level; it must be retrieved before any other objects. Similarly, a Session object must be retrieved before a Screen object.

OLE (Object Linking and Embedding) Automation: A Microsoft Windows standard for communications between applications. Windows applications that support this standard expose (make available) programmable objects that can be accessed by programming and macro languages. You can therefore create custom applications that use the built-in functionality of objects, which remain external to your applications. For example, instead of writing code to access mainframe data, you can have *EXTRA!* Personal Client objects do it for you.

Parent Object: An object that contains other objects in a hierarchical relationship. An object contained by a parent is known as a sub-object. Setting properties or performing methods on a parent object affects its sub-objects. The System objects is the parent of all *EXTRA!* Personal Client objects.

Presentation Space: An area in PC memory that stores the screen data of a display session. A presentation space includes data from the status line (called Operator Information Area in 3270 sessions). Each presentation space position stores a screen character. Using the methods and properties of Screen or Area objects, you can read from or write to the presentation space.

Property: A unique attribute of an *EXTRA!* Personal Client object that you set or return. For example, the expression *rc*=**Session.Connected** returns a value indicating whether the session was connected to the host.

QuickPad: A configurable secondary window from which you can execute *EXTRA!* Personal Client commands, macros, and host functions. You can use multiple Quickpads with a session. The settings defining a QuickPad are stored in a file with an EQP extension, located in the Schemes folder by default.

Xclock: A message that appears in the Operator Information Area (OIA) of a 3270 session screen, indicating that the mainframe is busy processing your input.