

Aviva® ActiveX® Automation
Objects Reference Manual



Aviva is a registered trademark of Aviva Solutions Inc.

Eicon is a registered trademark of Eicon Networks Corporation.

IBM is a registered trademark of International Business Machines Corporation.

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

VT220 is a trademark of Digital Equipment Corporation.

Proginet and IND\$FILE+ are trademarks of Proginet Corporation.

Portions copyright © 1992-1996 Summit Software Company.

All other trademarks and registered trademarks are the property of their respective owners.

Changes are periodically made to the information in this document. These changes will be incorporated into new editions of this document. Aviva Solutions Inc. may also make improvements and/or changes in the products and/or programs described in this document at any time.

Please forward your comments about this publication to:

support@avivasolutions.com

or

Aviva Solutions Inc.
Attention: Technical Publications
185 Dorval Avenue
Suite 303
Montreal, Quebec, Canada
H9S 5J9

Aviva Solutions Inc. may use or distribute whatever information you supply in any way it deems appropriate, without incurring any obligations to you.

Copyright © 1996-2016 Aviva Solutions Inc. All rights reserved, including those to reproduce this publication or parts thereof in any form without permission in writing from Aviva Solutions Inc.

Table of Contents

INTRODUCTION	11
AVIVA ACTIVEX AUTOMATION OVERVIEW	12
ABOUT AVIVABASIC	12
USING AVIVA'S TYPE LIBRARIES	13
USING DUAL INTERFACE INSTEAD OF IDISPATCH	13
AVIVABASIC PREDEFINED AUTOMATION OBJECTS AND DATA TYPES.....	13
USING ACTIVE X AUTOMATION TO OPEN OR CLOSE AVIVA SESSIONS	15
MULTIPLE CONTROLLERS INTERACTING WITH ONE SESSION.....	16
BUSY HOSTS.....	16
USING THE SETSHARING COMMAND TO CONTROL AN AVIVA SESSION.....	16
ARRAYS.....	17
COMMENTS.....	19
COMPARISON OPERATORS.....	19
CONSTANTS	21
ERROR HANDLING.....	25
EXPRESSION EVALUATION	26
KEYWORDS.....	28
LINE NUMBERS.....	29
LITERALS.....	29
NAMED PARAMETERS.....	30
OBJECTS	31
OPERATOR PRECEDENCE.....	33
OPERATOR PRECISION	34
USER-DEFINED TYPES.....	35
APPLICATION OBJECT.....	37
CLOSEALL (METHOD).....	38
EICONCONFIGDIR (METHOD)	39
EICONSYSTEMDIR (METHOD).....	40
EXECUTIONDIR (METHOD)	40
LASTERROR (METHOD)	41
LASTERRORMESSAGE (METHOD).....	42
LOGFILE (METHOD).....	43
MACRODIR (METHOD).....	43
OPENWORKSPACEFILE (METHOD).....	44
SAVeworkspaceAs (METHOD)	46
SESSIONDIR (METHOD).....	47
SESSIONINFOS (METHOD)	47
SESSIONS (METHOD).....	48

TRACEFILE (METHOD).....	49
WORKSPACE DIR (METHOD).....	50

APPWIN OBJECT..... 52

COPY (METHOD).....	53
DECREASEFONT SIZE (METHOD).....	54
DESELECT (METHOD).....	55
DISPLAYMODE (PROPERTY).....	55
FONT S (METHOD).....	56
GETPOSITION (METHOD).....	57
GETSIZE (METHOD).....	58
HOSTCOLOR (PROPERTY).....	59
INCREASEFONT SIZE (METHOD).....	60
PASTE (METHOD).....	60
PRINTSCREEN (METHOD).....	61
SAVESCREEN (METHOD).....	62
SELECT (METHOD).....	63
SELECTALL (METHOD).....	64
SETPOSITION (METHOD).....	65
SETSIZE (METHOD).....	66
SHOWSTATE (PROPERTY).....	67
VISIBLE (PROPERTY).....	68

COORDINATE OBJECT..... 69

X (PROPERTY).....	69
Y (PROPERTY).....	70

DESTINATION OBJECT..... 72

DESTGENERIC INTERFACE.....	72
DEST32TN INTERFACE.....	73
CONNECTIONTYPE (PROPERTY).....	73
DESTINATIONNAME (PROPERTY).....	74
DEVICENAME (PROPERTY).....	74
HOSTNAME (PROPERTY).....	75
PORTNUMBER (PROPERTY).....	75
SECURITY (PROPERTY).....	75

DESTINATIONS OBJECT 77

COUNT (PROPERTY).....	77
ITEM (METHOD).....	78
DESTINATIONINUSE (PROPERTY).....	78

FIELD OBJECT.....	79
ATTRIBUTE (PROPERTY)	80
GETDATA (METHOD).....	81
ISPROTECTED (PROPERTY)	82
LENGTH (PROPERTY)	83
NEXT (METHOD)	84
NEXTPROTECTED (METHOD)	85
NEXTUNPROTECTED (METHOD)	86
POSITION (PROPERTY)	87
PREV (METHOD).....	88
PREVPROTECTED (METHOD).....	89
PREVUNPROTECTED (METHOD)	90
SETDATA (METHOD).....	91
FILETRANSFER OBJECT.....	93
ABORT (METHOD).....	94
APPEND (PROPERTY)	95
BLOCKSIZE (PROPERTY).....	96
BYTESTRANSFERRED (METHOD)	96
CR LF (PROPERTY)	97
HOSTFILE (PROPERTY)	98
LOGRECLen (PROPERTY).....	99
ONFILETRANSFERDONE.....	99
PACKETSIZE (PROPERTY)	100
PCFILE (PROPERTY)	101
RECEIVE (METHOD).....	102
RECFORMAT (PROPERTY).....	103
RESET (METHOD).....	104
SCHEME (PROPERTY).....	105
SEND (METHOD)	106
SPACEALLOC (PROPERTY).....	107
SPACEINCREMENT (PROPERTY)	108
SPACEQUANTITY (PROPERTY).....	109
STATUS (METHOD).....	110
TIMEOUT (PROPERTY).....	111
TRANSLATE (PROPERTY)	112
USEROPTIONS (PROPERTY).....	113
FONT OBJECT	114
APPLY (METHOD)	115
NAME (PROPERTY)	116
SIZE (PROPERTY)	116
WIDTH (PROPERTY)	117

FONTS OBJECT	118
COUNT (PROPERTY).....	119
CURRENT (METHOD).....	120
ITEM (METHOD).....	121
HOSTBROWSER OBJECT	122
CANCEL (METHOD).....	123
HOSTFILES (METHOD)	124
MASK (PROPERTY).....	125
MAXCOUNT (PROPERTY).....	126
RESET (METHOD).....	127
STATUS (METHOD).....	127
TIMEOUT (PROPERTY).....	128
HOSTCOLOR OBJECT	130
APPLY (METHOD)	132
BLACK (PROPERTY).....	133
BLUE (PROPERTY).....	134
BROWN (PROPERTY).....	134
DARKGRAY (PROPERTY)	135
DEEPBLUE (PROPERTY).....	136
GRAY (PROPERTY).....	136
GREEN (PROPERTY)	137
ORANGE (PROPERTY).....	138
PALEGREEN (PROPERTY).....	138
PALETURQUOISE (PROPERTY)	139
PINK (PROPERTY).....	140
PURPLE (PROPERTY)	140
RED (PROPERTY).....	141
RGB (FUNCTION)	142
TURQUOISE (PROPERTY).....	144
WHITE (PROPERTY)	144
YELLOW (PROPERTY)	145
HOSTFILE OBJECT	146
ATTRIBUTE (PROPERTY)	147
NAME (PROPERTY)	148
PRINTFILE (FUNCTION).....	149
TIMESTAMP (PROPERTY).....	149
HOSTFILES OBJECT	151
COUNT (PROPERTY).....	152
ITEM (METHOD).....	153

HOTSPOT OBJECT.....	155
ACTIVE (PROPERTY)	156
ASBUTTON (PROPERTY)	156
MATCHCASE (PROPERTY)	157
NAME (PROPERTY)	158
HOTSPOTS OBJECT	159
COUNT (PROPERTY).....	160
ITEM (METHOD).....	160
LOAD (METHOD).....	161
REMOVE (METHOD).....	162
OIA OBJECT.....	164
APLMODE (PROPERTY).....	165
COMMCHK (PROPERTY)	166
GRAPHICCURSORMODE (PROPERTY)	167
INPUTINHIBIT (PROPERTY).....	167
INPUTINHIBITSTATE (PROPERTY)	168
INSERTMODE (PROPERTY).....	169
MACHINECHECK (PROPERTY)	170
MESSAGEWAITING (PROPERTY)	170
OWNERSHIP (PROPERTY)	171
PROGRAMCHECK (PROPERTY).....	172
SYSTEMAVAILABLE (PROPERTY).....	173
PRINTJOB OBJECT	174
CANCEL (METHOD).....	174
PA1 (METHOD).....	175
PA2 (METHOD)	176
PAUSE (METHOD).....	177
RESUME (METHOD).....	178
STATE (METHOD).....	179
PS OBJECT	180
ATTRIB (PROPERTY)	182
EXTENDEDATTRIB (PROPERTY).....	183
FIELD (METHOD).....	184
FINDSTRING (METHOD)	185
GETCURSORLOCATION (METHOD)	186
GETDATA (METHOD).....	188
MAXROWCOLUMN (METHOD).....	190
NULLTOSPACE (PROPERTY)	191
OIA (PROPERTY)	192

QUERYHOSTUPDATE (METHOD)	193
QUERYHOSTUPDATEWITHWAIT (METHOD)	194
RESET (METHOD).....	195
RETRIEVEKEY (METHOD).....	196
ROWCOLTOPPOSITION (METHOD)	197
SENDSTRING (METHOD)	198
SENDSTRINGTIMEOUT (PROPERTY).....	201
SETCURSORLOCATION (METHOD).....	201
SETDATA (METHOD).....	203
STARTHOSTNOTIFICATION (METHOD)	204
STARTKEYINTERCEPT (METHOD).....	205
STOPHOSTNOTIFICATION (METHOD).....	207
STOPKEYINTERCEPT (METHOD)	207
SUBSTITUTECHAR (PROPERTY)	208
WAITCURSORAT (METHOD).....	209
WAITCURSORMOVE (METHOD).....	211
WAITFORSTRING (METHOD)	213
WAITHOSTSETTLE (METHOD).....	214

ROWCOL OBJECT..... 216

COLUMN (PROPERTY)	217
ROW (PROPERTY)	217

SCREENTRIGGER OBJECT..... 219

ACTIVE (PROPERTY)	220
MATCHCASE (PROPERTY)	220
NAME (PROPERTY)	221
NOTIFY (PROPERTY)	222

SCREENTRIGGERS OBJECT 223

COUNT (PROPERTY).....	224
ITEM (METHOD).....	224
LOAD (METHOD).....	225
REMOVE (METHOD).....	226

SESSION OBJECT 228

ACTIVATE (METHOD).....	230
APPWIN (PROPERTY).....	231
BLOCKCLOSE (METHOD).....	231
BLOCKPLAY (METHOD).....	232
BLOCKUSERINPUT (METHOD)	233
CLOSE (METHOD)	234
CONNECT (METHOD)	235

CONNECTSSH(METHOD)	237
CONNECTIONSTATE (METHOD)	238
DESTINATIONS (METHOD)	239
DISCONNECT (METHOD)	240
FILETRANSFER (PROPERTY)	241
GETPROPERTIES (METHOD)	242
HOSTBROWSER (PROPERTY).....	243
HOTSPOTS (METHOD)	245
ISMACRORUNNING (METHOD)	245
LASTERROR (METHOD)	246
OLETRACE (PROPERTY).....	247
ONCONNECTIONFAIL	248
PRINTERNAME (PROPERTY).....	248
PRINTJOB (PROPERTY).....	249
PS (PROPERTY)	250
QUERYBLOCKCLOSE (METHOD)	251
RESET (METHOD).....	252
RESETCONNECTPARAM (METHOD)	253
RUNMACRO (METHOD)	254
SCREENTRIGGERS (METHOD)	255
SESSIONINFO (PROPERTY)	256
SETCONNECTPARAM (METHOD).....	257
SETPROPERTIES (METHOD).....	258
SETSHARING (METHOD)	260
UNBLOCKCLOSE (METHOD)	263
UNBLOCKPLAY (METHOD)	264
UNBLOCKUSERINPUT (METHOD).....	265

SESSIONS OBJECT **267**

ADD (METHOD).....	268
ARRANGE (METHOD)	269
CONTEXTFILTER (PROPERTY).....	271
COUNT (PROPERTY).....	271
ITEM (METHOD)	272
REMOVE (METHOD)	273

SESSIONINFO OBJECT **275**

CONNECTIONSTATE (PROPERTY).....	276
CONTEXT (PROPERTY)	277
FULLNAME (PROPERTY)	278
NAME (PROPERTY)	279
OWNER (PROPERTY)	280
OWNERID (PROPERTY).....	280
SHORTNAME (PROPERTY).....	280
STATE (PROPERTY)	281

TYPE (PROPERTY).....	282
SESSIONINFOS OBJECT	284
CONTEXTFILTER (PROPERTY).....	285
COUNT (PROPERTY).....	285
ITEM (METHOD).....	286
AVIVABASIC ERROR VALUES	288
APPENDIX: SESSION.SETCONNECTPARAM METHOD - PARAMETER VALUES	294
3270 DISPLAY PARAMETERS.....	294
3270 PRINTER PARAMETERS	298
5250 DISPLAY PARAMETERS.....	302
5250 PRINTER PARAMETERS	305
VT/TELNET DISPLAY PARAMETERS.....	309
VT/SSH DISPLAY PARAMETERS	312
INDEX.....	315

Introduction

The Aviva ActiveX[®] Automation Objects Reference Manual is a printable programmer's guide that provides you with information about ActiveX Automation objects exposed by Aviva for Desktops.

In this Reference Manual, you will find the following information:

Aviva ActiveX Automation overview: What is AvivaBasic and what it can be used for, how to use Aviva's type libraries, information about predefined automation objects and data types, and many more details about the structure and the syntax of ActiveX Automation objects and the AvivaBasic language. For more information, see [Aviva ActiveX Automation Overview](#) on page 12.

AvivaBasic objects, object collections, methods and properties.

AvivaBasic error values: A list of values returned by AvivaBasic when an error occurs in your macro script. For more information, see [AvivaBasic Error Values](#) on page 288.

The Aviva ActiveX Automation Objects Reference Manual does not include the entire contents of the Aviva Programmer's Reference. For more information about:

- the AvivaBasic dialog editor
- the AvivaBasic macro editor
- HLLAPI and WinHLLAPI

consult the online Programmer's Reference (on the **Help** menu of any display or printer session, click **Programmer's Reference**).

Aviva ActiveX Automation Overview

Aviva exposes ActiveX Automation objects that let you directly control, manipulate and interact with Aviva sessions as follows:

- Control an Aviva 3270, 5250, or VT220™ (hereafter referred to as VT) session from a VBA, native Windows or .NET application.
- Control a VBA, native Windows or .NET application from an Aviva 3270, 5250, or VT session.

For more information on, or how to use ActiveX Automation, refer to Microsoft's web site, any VBA online help for Microsoft Office products, or Microsoft Visual Studio documentation.

About AvivaBasic

AvivaBasic is a Visual Basic for Applications (VBA) compliant macro language that includes a dialog editor and macro debugger. Aviva shares with other Microsoft Office applications the ability to be used as an ActiveX Automation server or client and to control, or be controlled by other applications. Automation is integrated with AvivaBasic, giving you a familiar programming environment that makes designing your applications easier.

Before you start using AvivaBasic, review **About Macros** in the Aviva Help Topics, Microsoft Visual Studio documentation, or the VBA language reference found in any Microsoft Office Products.

You can use AvivaBasic to automate Aviva as follows:

- Record a macro
This is the easiest way to create code. View, edit and alter your code with Aviva's macro editor.
- Edit an existing macro
Open the macro editor to create, add or change macro (.evb) code.
- Use VBA
Write code using a Visual Basic compatible application with AvivaBasic's ActiveX automation objects.
- Use C++
Write code in C++ to interact with AvivaBasic's ActiveX automation objects.
- Use .NET languages
Write code in any language that supports .NET CLR (Common Language Runtime), such as C# or VB.NET to interact with AvivaBasic's ActiveX automation objects.

Other AvivaBasic features:

- Macro editor, debugger and dialog editor.
- Online help for the AvivaBasic language including code samples.
- Aviva ActiveX automation with various object extensions.

The online Programmer's Reference is a complete guide to the functions, definitions and operations of the AvivaBasic macro language.

Using Aviva's Type Libraries

This topic includes a brief description of a type library, and describes how to use the Aviva's type libraries in your application programs. If your application does not support type libraries, then refer to the online Programmer's Reference (AvivaBasic) for all the information that you need to get started with Aviva's Automation objects.

What is a type library?

A type library (.TLB) is a file, or a component within a file, that contains ActiveX (OLE) Automation descriptions of exposed objects, properties, and methods. If your programming tools support ActiveX Automation, you can use a type library to access the exposed objects of another application. Many Microsoft products, including Visual Studio and Office (VBA) support type libraries.

Aviva includes theAVIVA.TLB type library. This file provides you with an object model for Aviva objects and the syntax for Aviva methods and properties.

Using Dual Interface instead of IDispatch

Aviva Automation code examples use syntax that demonstrate IDispatch technology. This topic explains the advantage that you gain by using Aviva's Dual Interface technology in your code instead of IDispatch.

Using IDispatch or Dual Interface

Your applications can access Aviva objects through IDispatch or Dual Interface. The IDispatch interface uses late binding which occurs at run time. You start Aviva using IDispatch as follows:

```
Dim MyAviva As Object  
Set MyAviva = CreateObject("Aviva.Application")
```

The Dual Interface uses early binding which occurs at compile time, and runs faster than IDispatch interface. You start Aviva using Dual Interface as follows:

```
Dim MyAviva As CAviva  
Set MyAviva = CreateObject("Aviva.Application")
```

Note Aviva's macro editor only supports IDispatch syntax.

See Also

[Using Aviva's Type Libraries](#) on page 13

AvivaBasic Predefined Automation Objects and Data Types

Aviva exposes ActiveX Automation objects that let you directly control, manipulate and interact with Aviva sessions. AvivaBasic macros include built-in equivalents (predefined) for most of Aviva's Automation objects. For example, in an Aviva AvivaBasic macro, the FileTransfer object is declared by using the predefined

statement, FileTransfer, or is declared by using ActiveX Automation syntax. The following example shows both methods, and syntax used throughout the Programmer's reference.

Syntax

AvivaBasic Macro

```
rc& = FileTransfer.BytesTransferred
```

ActiveX client application

```
rc& = Object1.BytesTransferred
```

(where Object1 is declared in a DIM statement)

ActiveX Automation syntax

Objects that are not predefined must be created and manipulated by using ActiveX Automation syntax. In the online Programmer's Reference, these objects are identified as follows:

"This method or property may only be executed from an ActiveX controller application."

Predefined data types

AvivaBasic also includes predefined data types that let you optimize your code. However, these data types can only be used in AvivaBasic macro scripts.

The following is a list of predefined Automation objects included in AvivaBasic. For detailed information on these objects, consult the online Programmer's Reference.

Predefined Objects
Application
AppWin or Session.AppWin
FileTransfer or Session.FileTransfer (3270 only)
Font
HostColor or AppWin.HostColor (3270 and 5250 only)
OIA or PS.OIA (3270 and 5250 only)
PS or Session.PS
Session
SessionInfo or Session.SessionInfo
Predefined Collection Objects
Fonts or AppWin.Fonts

Note All collection objects return a collection of objects. For example, the Fonts object returns a collection of Font objects.

The following is a list of predefined data types included in AvivaBasic. For detailed information on these data types, consult the online Programmer's Reference.

Predefined Data Types
Coord
Field (3270 and 5250 only)
Font
RowCol

Using ActiveX Automation to Open or Close Aviva Sessions

In your AvivaBasic macros, or VBA compliant application, you can open or close an Aviva host session by using the `GetObject` or `CreateObject` command.

There are Aviva ActiveX Automation objects that are predefined in AvivaBasic. For an explanation and listing of these predefined objects, refer to [AvivaBasic Predefined Automation Objects and Data Types](#) on page 13. You should also browse our code examples to see how to use Aviva's Automation objects, methods and properties.

Open an Aviva session

You can open, start, and then manipulate an Aviva session by using the following methods, AvivaBasic macro, or ActiveX Automation.

AvivaBasic Macro

To start your session, double-click on a session file (.A3D, .A5D, or .AVD). In the AvivaBasic macro editor, this session can then be manipulated by using predefined objects, `Application`, `Session`, or `SessionInfo`.

ActiveX Automation (VBA and AvivaBasic)

Use the `CreateObject` function to start an Aviva `Application` object, and then add your session by using the `Sessions.Add` method.

With the Aviva `Application` object, you get complete control of an Aviva workspace, including all Automation collections, methods and properties. For example, the `Application` and `Sessions` object have methods to manipulate any session, or a collection of sessions.

Use the `GetObject` function to start an Aviva session. However, by using this method, you can only manipulate Aviva session methods and properties. You code these functions as follows:

CreateObject

```
Dim MySession As Object
Dim MySession As Object

Set AvivaApp = CreateObject("Aviva.Application")
Set MySession = AvivaApp.Sessions.Add("MySession.a3d", True)
```

GetObject

```
Dim MySession As Object
Set MySession = GetObject("MySession.a3d")
```

Close an Aviva host session

You can use any of the following methods to close your Aviva sessions:

- The `Application` object to close all sessions.
- The `Sessions` object to remove a session.
- The `Session` object to close a session.

Multiple Controllers Interacting with One Session

Aviva's Application object allows several controllers to communicate with a display session by preventing it from being added to a workspace more than once. This control is necessary to avoid any problems that could occur when, for example, two ActiveX clients or AvivaBasic macro scripts attempt to send keystrokes to the same session.

Limiting access to a session with display object

By default, the first ActiveX program or macro script to execute the SetSharing method on a session is registered as the "owner" of that session.

The SetSharing method parameter, shareRight%, can have different values, and access to a Session is given according to this parameter.

For a complete list of available shareRight% options, and detailed descriptions of what each one allows a script to do, refer to the topic [SetSharing \(method\)](#) on page 260.

See Also

[Using the SetSharing Command to Control an Aviva Session](#) on page 16

Busy Hosts

When you execute methods that cause Aviva sessions to interact with hosts, you must consider the possibility that your host may be busy. Your program may receive return codes other than 0 (zero) and these error codes result from the inability of a host to immediately respond to session input.

Tip

Whenever a method involving host interactions returns a result code other than 0 (zero), you should have your program pause for a brief period. This allows the host to get ready, at which time you can try the following:

- Execute the method again.
- Retry executing the method more than once.
- Use the PS.WaitHostSettle method
- Use the PS.WaitForString method

See Also

[PS Object](#) on page 180
[OIA Object](#) on page 164

Using the SetSharing Command to Control an Aviva Session

Multiple applications of various types (ActiveX controllers, HLLAPI applications and AvivaBasic macros) can share access to the same session. AvivaBasic macros have a higher access priority to Aviva sessions than ActiveX controllers and HLLAPI

applications. ActiveX controllers and HLLAPI applications are considered external applications, and share a session on a first-come, first-serve basis. For more information about sharing rights, refer to the description of the SetSharing command.

Difference between AvivaBasic macros and external applications

ActiveX controllers must issue a SetSharing command with the desired sharing rights to a specified session. Controller and HLLAPI applications have the same restrictions imposed by the SetSharing mechanism. However, AvivaBasic macros do not have this restriction, and always run. Also, macros are not required to use SetSharing as a prerequisite call for macro commands.

However, if you want your AvivaBasic macro to have full control of a session, and not allow any external applications to intervene, then your macro should use SetSharing with NO_SHARING. This action also stops other macros from successfully using a SetSharing command.

If you want a controller or HLLAPI application to prevent a macro from running, you must use BlockPlay or a Reserve HLLAPI function call.

Arrays

Declaring Array Variables

Arrays in AvivaBasic are declared using any of the following statements:

```
Dim  
Public  
Private
```

For example:

```
Dim a(10) As Integer  
Public LastNames(1 to 5,-2 to 7) As Variant  
Private FirstNames(1 to 5,-2 to 7) As Variant
```

Arrays of any data type can be created, including Integer, Long, Single, Double, Boolean, Date, Variant, Object, user-defined structures, and data objects.

The lower and upper bounds of each array dimension must be within the following range:

```
-32768 <= bound <= 32767
```

Arrays can have up to 60 dimensions.

Arrays can be declared as either fixed or dynamic, as described below.

Fixed Arrays

The dimensions of fixed arrays cannot be adjusted at execution time. Once declared, a fixed array will always require the same amount of storage. Fixed arrays can be declared with the Dim, Private, or Public statement by supplying explicit dimensions. The following example declares a fixed array of ten strings.

```
Dim a(10) As String
```

Fixed arrays can be used as members of user-defined data types. The following example shows a structure containing fixed-length arrays.

```
Type Foo
rect(4) As Integer
colors(10) As Integer
End Type
```

Only fixed arrays can appear within structures.

Dynamic Arrays

Dynamic arrays are declared without explicit dimensions, as shown below.

```
Public Ages() As Integer
```

Dynamic arrays can be resized at execution time using the Redim statement. For example:

```
ReDim Ages$(100)
```

Subsequent to their initial declaration, dynamic arrays can be re-dimensioned any number of times. When re-dimensioning an array, the old array is first erased unless you use the Preserve keyword, as shown below:

```
ReDim Preserve Ages$(100)
```

Dynamic arrays cannot be members of user-defined data types.

Passing Arrays

Arrays are always passed by reference. When you pass an array, you can specify the array name by itself, or with parentheses as shown below.

```
Dim a(10) As String
FileList 'Both of these are OK
FileList a()
```

Querying Arrays

The following table describes the functions used to retrieve information about arrays.

Use this function	To do this
LBound	Retrieve the lower bound of an array. A runtime is generated if the array has no dimensions.
UBound	Retrieve the upper bound of an array. A runtime error is generated if the array has no dimensions.
ArrayDims	Retrieve the number of dimensions of an array. This function returns 0 if the array has no dimensions.

Operations on Arrays

The following table describes the functions that operate on arrays.

Use this command	To do this
ArraySort	Sort an array of integers, longs, singles, doubles, currency, Booleans, dates, or variants.
FileList	Fill an array with a list of files in a given directory.
DiskDrives	Fill an array with a list of valid drive letters.
AppList	Fill an array with a list of running applications.

WinList	Fill an array with a list of top-level windows.
SelectBox	Display the contents of an array in a list box.
PopupMenu	Display the contents of an array in a popup menu.
ReadInSection	Fill an array with the item names from a section in an INI file.
FileDirs	Fill an array with a list of subdirectories.
Erase	Erase all the elements of an array.
ReDim	Establish the bounds and dimensions of an array.
Dim	Declare an array.

Comments

Comments can be added to AvivaBasic code in the following manner:

All text between a single quotation mark and the end of the line is ignored:

```
MsgBox "Hello"           'Displays a message box.
```

The REM statement causes the compiler to ignore the entire line:

```
REM This is a comment.
```

AvivaBasic supports C-style multiline comment blocks `/*...*/`, as shown in the following example:

```
Sub Main()
  MsgBox "Before comment"
  /* This stuff is all commented out.
  This line, too, will be ignored.
  This is the last line of the comment. */
  MsgBox "After comment"
End Sub
```

Note C-style comments can be nested.

Comparison Operators

Syntax

```
expression1 [< | > | <= | >= | <> | =] expression2
```

Description

Comparison operators return True or False depending on the operator.

Comments

The comparison operators are listed in the following table:

Operator	Returns True If
>	expression1 is greater than expression2
<	expression1 is less than expression2

<=	expression1 is less than or equal to expression2
>=	expression1 is greater than or equal to expression2
<>	expression1 is not equal to expression2
=	expression1 is equal to expression2

The comparison operators behave differently depending on the combination and type of expressions, as shown in the following table:

If one expression is	and the other expression is	Then
Numeric	Numeric	A numeric comparison is performed.
String	String	A string comparison is performed.
Numeric	String	A compile error is generated.
Variant	String	A string comparison is performed.
Variant	Numeric	A variant comparison is performed.
Null variant	Any data type	Returns Null.
Variant	Variant	A variant comparison is performed.

String Comparisons

If the two expressions are strings, then the operator performs a text comparison between the two string expressions, returning True if expression1 is less than expression2. The text comparison is case-sensitive if Option Compare is Binary; otherwise, the comparison is case-insensitive.

When comparing letters with regard to case, lowercase characters in a string sort greater than uppercase characters, so a comparison of "a" and "A" would indicate that "a" is greater than "A".

Numeric Comparisons

When comparing two numeric expressions, the less precise expression is converted to be the same type as the more precise expression.

Dates are compared as doubles. This may produce unexpected results as it is possible to have two dates that, when viewed as text, display as the same date when, in fact, they are different as shown in the following example:

Example

```
Sub Main()
Dim date1 As Date
Dim date2 As Date
    date1 = Now
    date2 = date1 + 0.000001      'Adds a fraction of a second.
    MsgBox date2 = date1        'Prints False (the dates are
                                different).
    MsgBox date1 & ", " & date2 'Prints two dates that are the
                                same.
End Sub
```

Variant Comparisons

When comparing variants, the actual operation performed is determined at execution time according to the following table:

If one variant is	And the other variant is	Then
Numeric	Numeric	Compares the variants as numbers.
String	String	Compares the variants as text.
Numeric	String	The number is less than the string.
Null	Any other data type	Null.
Numeric	Empty	Compares the number with 0.
String	Empty	Compares the string with a zero-length string.

Example

This example demonstrates the < and > operators.

```
Sub Main()
    'Test two numbers and display
    If 5 < 2 Then
        MsgBox "5 is less than 2."
    Else
        MsgBox "5 is not less than 2."
    End If
    'Test two strings and display the result
    If "This" < "That" Then
        MsgBox "'This' is less than 'That'."
    Else
        MsgBox "'That' is less than 'This'."
    End If
End Sub
```

See Also

[Operator Precedence](#) on page 33

Is (operator), Like (operator), and Option Compare (statement) in the online Programmer's Reference

Constants

Constants are variables that cannot change value during script execution. The following constants are predefined by AvivaBasic.

Constant	Value	Description
True	-1	Boolean value True.
True	True	
False	0	Boolean value False.
False	False	

Empty	Empty	Variant of type 0, indicating that the variant is uninitialized.
Nothing	0	Value indicating that an object variable no longer references a valid object.
Null	Null	Variant of type 1, indicating that the variant contains no data.
Null	Null	
Pi	3.1415...	Value of Pi.
Win32		True if development environment is 32-bit Windows.
Empty	Empty	
ebMinimized	1	The application is minimized.
ebMaximized	2	The application is maximized.
ebRestored	3	The application is restored.
ebCFText	1	Text.
ebCFBitmap	2	Bitmap
ebCFMetafile	3	Metafile
ebCFDIB	8	Device-independent bitmap.
ebCFPalette	9	Palette
ebCFUnicode	13	Unicode text.
ebUseSunday	0	Use the date setting as specified by the current locale.
ebSunday	1	Sunday
ebMonday	2	Monday
ebTuesday	3	Tuesday
ebWednesday	4	Wednesday
ebThursday	5	Thursday
ebFriday	6	Friday
ebSaturday	7	Saturday
ebFirstJan1	1	Start with week in which January 1 occurs.
ebFirstFourDays	2	Start with first week with at least four days in the new year.
ebFirstFullWeek	3	Start with first full week of the year.
ebNormal	0	Read-only, archive, subdir, and none.
ebReadOnly	1	Read-only files.
ebHidden	2	Hidden files.
ebSystem	4	System files.
ebVolume	8	Volume labels.
ebDirectory	16	Subdirectory

ebArchive	32	Files that have changed since the last backup.
ebNone	64	Files with no attributes.
ebDOS	1	A console executable file.
ebWindows	2	A Windows executable file.
ebRegular	1	Normal font (i.e., neither bold nor italic).
ebItalic	2	Italic font.
ebBold	4	Bold font.
ebBoldItalic	6	Bold-italic font.
ebIMENoOp	0	IME not installed.
ebIMEOn	1	IME on.
ebIMEOff	2	IME off.
ebIMEDisabled	3	IME disabled.
ebIMEHiragana	4	Hiragana double-byte character.
ebIMEKatakanaDbl	5	Katakana double-byte characters.
ebIMEKatakanaSng	6	Katakana single-byte characters.
ebIMEAlphaDbl	7	Alphanumeric double-byte characters.
ebIMEAlphaSng	8	Alphanumeric single-byte characters.
ebOKOnly	0	Displays only the OK button.
ebOKCancel	1	Displays OK and Cancel buttons.
ebAbortRetryIgnore	2	Displays Abort, Retry, and Ignore buttons.
ebYesNoCancel	3	Displays Yes, No, and Cancel buttons.
ebYesNo	4	Displays Yes and No buttons.
ebRetryCancel	5	Displays Cancel and Retry buttons.
ebCritical	16	Displays the stop icon.
ebQuestion	32	Displays the question icon.
ebExclamation	48	Displays the exclamation icon.
ebInformation	64	Displays the information icon.
ebApplicationModal	0	The current application is suspended until the dialog box is closed.
ebDefaultButton1	0	First button is the default button.
ebDefaultButton2	256	Second button is the default button.
ebDefaultButton3	512	Third button is the default button.
ebSystemModal	4096	All applications are suspended until the dialog box is closed.
ebOK	1	Returned from MsgBox indicating that OK was pressed.
ebCancel	2	Returned from MsgBox indicating that Cancel was pressed.

ebAbort	3	Returned from MsgBox indicating that Abort was pressed.
ebRetry	4	Returned from MsgBox indicating that Retry was pressed.
ebIgnore	5	Returned from MsgBox indicating that Ignore was pressed.
ebYes	6	Returned from MsgBox indicating that Yes was pressed.
ebNo	7	Returned from MsgBox indicating that No was pressed.
ebWin32	2	Microsoft Windows 32-bit
ebLandscape	1	Landscape paper orientation.
ebPortrait	2	Portrait paper orientation.
ebLeftButton	1	Left mouse button.
ebRightButton	2	Right mouse button.
ebHide	0	Application is initially hidden.
ebNormalFocus	1	Application is displayed at the default position and has the focus.
ebMinimizedFocus	2	Application is initially minimized and has the focus.
ebMaximizedFocus	3	Application is maximized and has the focus.
ebNormalNoFocus	4	Application is displayed at the default position and does not have the focus.
ebMinimizedNoFocus	6	Application is minimized and does not have the focus.
ebUpperCase	1	Converts string to uppercase.
ebLowerCase	2	Converts string to lowercase.
ebProperCase	3	Capitalizes the first letter of each word.
ebWide	4	Converts narrow characters to wide characters.
ebNarrow	8	Converts wide characters to narrow characters.
ebKatakana	16	Converts Hiragana characters to Katakana characters.
ebHiragana	32	Converts Katakana characters to Hiragana characters.
ebUnicode	64	Converts string from MBCS to UNICODE.
ebFromUnicode	128	Converts string from UNICODE to MBCS.
ebEmpty	0	Variant has not been initialized.
ebNull	1	Variant contains no valid data.
ebInteger	2	Variant contains an Integer.

ebLong	3	Variant contains a Long.
ebSingle	4	Variant contains a Single.
ebDouble	5	Variant contains a Double.
ebCurrency	6	Variant contains a Currency.
ebDate	7	Variant contains a Date.
ebString	8	Variant contains a String.
ebObject	9	Variant contains an Object.
ebError	10	Variant contains an Error.
ebBoolean	11	Variant contains a Boolean.
ebVariant	12	Variant contains an array of Variants.
ebDataObject	13	Variant contains a data object.
ebArray	8192	Added to any of the other types to indicate an array of that type.
ebBack	Chr\$(8)	String containing a backspace.
ebCr	Chr\$(13)	String containing a carriage return.
ebCrLf	Chr\$(13) and Chr\$(10)	String containing a carriage-return linefeed pair.
ebFormFeed	Chr\$(11)	String containing a form feed.
ebLf	Chr\$(10)	String containing a line feed.
ebNullChar	Chr\$(0)	String containing a single null character.
ebNullString	0	Special string value used to pass null pointers to external routines.
ebTab	Chr\$(9)	String containing a tab.
ebVerticalTab	Chr\$(12)	String containing a vertical tab.

You can define your own constants using the Const statement. Preprocessor constants are defined using #Const.

Error Handling

Error Handlers

AvivaBasic supports nested error handlers. When an error occurs within a subroutine, AvivaBasic checks for an **On Error** handler within the currently executing subroutine or function. An error handler is defined as follows:

```
Sub foo()
    On Error Goto catch
    'Do something here.
Exit Sub
catch:
    'Handle error here.
End Sub
```

Error handlers have a life local to the procedure in which they are defined. The error is reset when any of the following conditions occurs:

- An **On Error** or **Resume** statement is encountered.
- When **Err.Number** is set to -1.
- When the **Err.Clear** method is called.
- When an **Exit Sub**, **Exit Function**, **End Function**, **End Sub** is encountered.

Cascading Errors

If a runtime error occurs and no **On Error** handler is defined within the currently executing procedure, then AvivaBasic returns to the calling procedure and executes the error handler there. This process repeats until a procedure is found that contains an error handler or until there are no more procedures. If an error is not trapped or if an error occurs within the error handler, then AvivaBasic displays an error message, halting execution of the script.

Once an error handler has control, it should address the condition that caused the error and resume execution with the **Resume** statement. This statement resets the error handler, transferring execution to an appropriate place within the current procedure. The error is reset if the procedure exits without first executing **Resume**.

Visual Basic Compatibility

Where possible, AvivaBasic has the same error numbers and error messages as Visual Basic. This is useful for porting scripts between environments.

Handling errors in AvivaBasic involves querying the error number or error text using the **Error\$** function or **Err.Description** property. Since this is the only way to handle errors in AvivaBasic, compatibility with Visual Basic's error numbers and messages is essential.

AvivaBasic errors fall into three categories:

1. Visual Basic-compatible errors: These errors, numbered between 0 and 799, are numbered and named according to the errors supported by Visual Basic.
2. AvivaBasic errors: These errors, numbered from 800 to 999, are unique to AvivaBasic.
3. User-defined errors: These errors, equal to or greater than 1,000, are available for use by extensions or by the script itself.

You can intercept trappable errors using AvivaBasic's **On Error** construct. Almost all errors in AvivaBasic are trappable except for various system errors.

Expression Evaluation

AvivaBasic allows expressions to involve data of different types. When this occurs, the two arguments are converted to be of the same type by promoting the less precise operand to the same type as the more precise operand. For example, AvivaBasic will promote the value of `i%` to a Double in the following expression:

```
result# = i% * d#
```

In some cases, the data type to which each operand is promoted is different than that of the most precise operand. This is dependent on the operator and the data types of the two operands and is noted in the description of each operator.

If an operation is performed between a numeric expression and a String expression, then the String expression is usually converted to be of the same type as the numeric expression. For example, the following expression converts the String expression to an Integer before performing the multiplication:

```
result = 10 * "2"           'Result is equal to 20.
```

There are exceptions to this rule, as noted in the description of the individual operators.

Type Coercion

AvivaBasic performs numeric type conversion automatically. Automatic conversions sometimes result in overflow errors, as shown in the following example:

```
d# = 45354
i% = d#
```

In this example, an overflow error is generated because the value contained in d# is larger than the maximum size of an Integer.

Rounding

When floating-point values (Single or Double) are converted to integer values (Integer or Long), the fractional part of the floating-point number is lost, rounding to the nearest integer value. AvivaBasic uses Baker's rounding:

- If the fractional part is larger than .5, the number is rounded up.
- If the fractional part is smaller than .5, the number is rounded down.
- If the fractional part is equal to .5, then the number is rounded up if it is odd and down if it is even.

The following table shows sample values before and after rounding.

Before Rounding	After Rounding to Whole Number
2.1	2
2.5	2
3.5	4
4.6	5

Default Properties

When an ActiveX object variable or an Object variant is used with numerical operators such as addition or subtraction, then the default property of that object is automatically retrieved. For example, consider the following:

```
Dim Excel As Object
Set Excel = GetObject("Excel.Application")
MsgBox "This application is " & Excel
```

The above example displays "This application is Microsoft Excel" in a dialog box. When the variable Excel is used within the expression, the default property is automatically retrieved, which, in this case, is the string "Microsoft Excel." Considering that the default property of the Excel object is .Value, then the following two statements are equivalent:

```
MsgBox "This application is " & Excel
MsgBox "This application is " & Excel.Value
```

Keywords

A keyword is any word or symbol recognized by AvivaBasic as part of the language. All of the following are keywords:

Access	Alias	And	Any
Append	As	Base	Begin
Binary	Boolean	ByRef	ByVal
Call	CancelButton	Case	CDecl
CheckBox	Chr	ChrB	ChrW
Close	ComboBox	Compare	Const
CStrings	Currency	Date	Declare
Default	DefBool	DefCur	DefDate
DefDbl	DefInt	DefLng	DefObj
DefSng	DefStr	DefVar	Dialog
Dim	Do	Double	DropListBox
Else	Elseif	End	Eqv
Error	Exit	Explicit	For
Function	Get	Global	GoSub
Goto	GroupBox	HelpButton	If
Imp	Inline	Input	InputB
Integer	Is	Len	Let
Lib	Like	Line	ListBox
Lock	Long	Loop	LSet
Mid	MidB	Mod	Name
New	Next	Not	Nothing
Object	Off	OKButton	On
Open	Option	Optional	OptionButton
OptionGroup	Or	Output	ParamArray
Pascal	Picture	PictureButton	Preserve
Print	Private	Public	PushButton
Put	Random	Read	ReDim
Rem	Resume	Return	RSet
Seek	Select	Set	Shared
Single	Spc	Static	StdCall
Step	Stop	String	Sub
System	Tab	Text	TextBox
Then	Time	To	Type

Unlock	Until	Variant	WEnd
While	Width	Write	Xor

Restrictions

All keywords are reserved by AvivaBasic, in that you cannot create a variable, function, constant, or subroutine with the same name as a keyword. However, you are free to use all keywords as the names of structure members.

For all other keywords in AvivaBasic (such as MsgBox, Str, and so on), the following restrictions apply:

- You can create a subroutine or function with the same name as a keyword.
- You can create a variable with the same name as a keyword as long as the variable is first explicitly declared with a Dim, Private, or Public statement.

Line Numbers

AvivaBasic does not support line numbers. As an alternative to line numbers, you can use meaningful labels as targets for absolute jumps, as shown below:

```
Sub Main()
Dim i As Integer
    On Error Goto MyErrorTrap
    i = 0
LoopTop:
    i = i + 1
    If i < 10 Then Goto LoopTop
MyErrorTrap:
    MsgBox "An error occurred."
End Sub
```

Literals

Literals are values of a specific type. The following table shows the different types of literals supported by AvivaBasic.

Literal	Description
10	Integer whose value is 10.
43265	Long whose value is 43,265.
5#	Double whose value is 5.0. A number's type can be explicitly set using any of the following type-declaration characters:
% Integer	& Long
# Double	! Single
5.5	Double whose value is 5.5. Any number with decimal point is considered a double.

5.4E100	Double expressed in scientific notation.
&HFF	Integer expressed in hexadecimal.
&O47	Integer expressed in octal.
&HFF#	Double expressed in hexadecimal.
"hello"	String of five characters: hello.
""hello""	String of seven characters: "hello". Quotation marks can be embedded within strings by using two consecutive quotation marks.
#1/1/1994#	Date value whose internal representation is 34335.0. Any valid date can appear with #'s. Date literals are interpreted at execution time using the locale settings of the host environment. To ensure that date literals are correctly interpreted for all locales, use the international date format:YYYY-MM-DD HH:MM:SS#

Constant Folding

AvivaBasic supports constant folding where constant expressions are calculated by the compiler at compile time. For example, the expression

```
i% = 10 + 12
```

is the same as:

```
i% = 22
```

Similarly, with strings, the expression

```
s$ = "Hello," + " there" + Chr(46)
```

is the same as:

```
s$ = "Hello, there."
```

Named Parameters

Many language elements in AvivaBasic support named parameters. Named parameters allow you to specify parameters to a function or subroutine by name rather than in adherence to a predetermined order. The examples below show various calls to MsgBox both using parameter by both name and position.

- By Name: MsgBox Prompt:= "Hello, world."
- By Position: MsgBox "Hello, world."
- By Name: MsgBox Title:="Title", Prompt:="Hello, world."
- By Position: MsgBox "Hello, world",,"Title"
- By Name: MsgBox HelpFile:="BASIC.HLP", _
 Prompt:="Hello, world.", HelpContext:=10
- By Position: MsgBox "Hello, world.",,"BASIC.HLP",10

Using named parameter makes your code easier to read, while at the same time removes you from knowing the order of parameter. With function that require many parameters, most of which are optional (such as MsgBox), code becomes significantly easier to write and maintain.

When supported, the names of the named parameter appear in the description of that language element.

When using named parameter, you must observe the following rules:

- Named parameter must use the parameter name as specified in the description of that language element. Unrecognized parameter names cause compiler errors.
- All parameters, whether named or positional, are separated by commas.
- The parameter name and its associated value are separated with :=
- If one parameter is named, then all subsequent parameter must also be named as shown below:

```
MsgBox "Hello, world", Title:="Title"      'OK
MsgBox Prompt:="Hello, world.",, "Title"  'WRONG!!!
```

Objects

AvivaBasic defines two types of objects: data objects and ActiveX Automation objects. Syntactically, these are referenced in the same way.

What Is an Object?

An object in AvivaBasic is an encapsulation of data and routines into a single unit. The use of objects in AvivaBasic has the effect of grouping together a set of functions and data items that apply only to a specific object type.

Objects expose data items for programmability called properties. For example, a sheet object may expose an integer called NumColumns. Usually, properties can be both retrieved (get) and modified (set).

Objects also expose internal routines for programmability called methods. In AvivaBasic, an object method can take the form of a function or a subroutine. For example, a ActiveX Automation object called MyApp may contain a method subroutine called Open that takes a single argument (a filename), as shown below:

```
MyApp.Open "c:\files\sample.txt"
```

Declaring Object Variables

To gain access to an object, you must first declare an object variable using either Dim, Public, or Private. For example:

```
Dim o As Object           'ActiveX Automation object
```

Initially, objects are given the value 0 (or Nothing). Before an object can be accessed, it must be associated with a physical object.

Assigning a Value to an Object Variable

An object variable must reference a real physical object before accessing any properties or methods of that object. To instantiate an object, use the Set statement.

```
Dim MyApp As Object
Set MyApp = CreateObject("Server.Application")
```

Accessing Object Properties

Once an object variable has been declared and associated with a physical object, it can be modified using AvivaBasic code. Properties are syntactically accessible using the dot operator, which separates an object name from the property being accessed.

```
MyApp.BackgroundColor = 10
i% = MyApp.DocumentCount
```

Properties are set using AvivaBasic's normal assignment statement:

```
MyApp.BackgroundColor = 10
```

Object properties can be retrieved and used within expressions:

```
i% = MyApp.DocumentCount + 10
MsgBox "Number of documents = " & MyApp.DocumentCount
```

Accessing Object Methods

Like properties, methods are accessed via the dot operator. Object methods that do not return values behave like subroutines in AvivaBasic (i.e., the arguments are not enclosed within parentheses).

```
MyApp.Open "c:\files\sample.txt", True, 15
```

Object methods that return a value behave like function calls in AvivaBasic. Any arguments must be enclosed in parentheses:

```
If MyApp.DocumentCount = 0 Then MsgBox "No open documents."
NumDocs = app.count(4, 5)
```

There is no syntactic difference between calling a method function and retrieving a property value, as shown below:

```
variable = object.property(arg1, arg2)
variable = object.method(arg1, arg2)
```

Comparing Object Variables

The values used to represent objects are meaningless to the script in which they are used, with the following exceptions:

- Objects can be compared to each other to determine whether they refer to the same object.
- Objects can be compared with Nothing to determine whether the object variable refers to a valid object.

Object comparisons are accomplished using the Is operator:

```
If a Is b Then MsgBox "a and b are the same object."
If a Is Nothing Then MsgBox "a is not initialized."
If b Is Not Nothing Then MsgBox "b is in use."
```

Collections

A collection is a set of related object variables. Each element in the set is called a member and is accessed via an index, either numeric or text, as shown below:

```
MyApp.Toolbar.Buttons(0)  
MyApp.Toolbar.Buttons("Tuesday")
```

It is typical for collection indexes to begin with 0.

Each element of a collection is itself an object, as shown in the following examples:

```
Dim MyToolBarButton As Object  
Set MyToolBarButton = MyApp.Toolbar.Buttons("Save")  
MyApp.Toolbar.Buttons(1).Caption = "Open"
```

The collection itself contains properties that provide you with information about the collection and methods that allow navigation within that collection:

```
Dim MyToolBarButton As Object  
NumButtons% = MyApp.Toolbar.Buttons.Count  
MyApp.Toolbar.Buttons.MoveNext  
MyApp.Toolbar.Buttons.FindNext "Save"  
For i = 1 To MyApp.Toolbar.Buttons.Count  
    Set MyToolBarButton = MyApp.Toolbar.Buttons(i)  
    MyToolBarButton.Caption = "Copy"  
Next i
```

Predefined Objects

AvivaBasic predefines a few objects for use in all scripts. These are:

Clipboard	System	Desktop	HWND
Net	Basic	Screen	

Note Some of these objects are not available on all platforms.

Operator Precedence

The following table shows the precedence of the operators supported by AvivaBasic. Operations involving operators of higher precedence occur before operations involving operators of lower precedence. When operators of equal precedence occur together, they are evaluated from left to right.

Operator	Description	Precedence Order
----------	-------------	------------------

()	Parentheses	Highest
^	Exponentiation	
-	Unary minus	
/, *	Division and multiplication	
\	Integer division	
Mod	Modulo	
+, -	Addition and subtraction	
&	String concatenation	
=, <>, >, <, <=, >=	Relational	
Like, Is	String and object comparison	
Not	Logical negation	
And	Logical or binary conjunction	
Or	Logical or binary disjunction	
Xor, Eqv, Imp	Logical or binary operators	

The precedence order can be controlled using parentheses, as shown below:

a = 4 + 3 * 2'	a becomes 10.
a = (4 + 3) * 2'	a becomes 14.

Operator Precision

When numeric, binary, logical or comparison operators are used, the data type of the result is generally the same as the data type of the more precise operand. For example, adding an Integer and a Long first converts the Integer operand to a Long, then performs a long addition, overflowing only if the result cannot be contained with a Long. The order of precision is shown in the following list.

Empty	Least precise
Boolean	
Integer	
Long	
Single	
Date	
Double	
Currency	

There are exceptions noted in the descriptions of each operator.

The rules for operand conversion are further complicated when an operator is used with variant data. In many cases, an overflow causes automatic promotion of the result to the next highest precise data type. For example, adding two Integer variants results in an Integer variant unless it overflows, in which case the result is automatically promoted to a Long variant.

User-Defined Types

User-defined types (UDTs) are structure definitions created using the Type statement. UDTs are equivalent to C language structures.

Declaring Structures

The Type statement is used to create a structure definition. Type declarations must appear outside the body of all subroutines and functions within a script and are therefore global to an entire script.

Once defined, a UDT can be used to declare variables of that type using the Dim, Public, or Private statement. The following example defines a rectangle structure.

```
Type Rect
    left As Integer
    top As Integer
    right As Integer
    bottom As Integer
End Type
:
Sub Main()
    Dim r As Rect
        :
        r.left = 10
    End Sub
```

Any fundamental data type can be used as a structure member, including other user-defined types. Only fixed arrays can be used within structures.

Copying Structures

UDTs of the same type can be assigned to each other, copying the contents. No other standard operators can be applied to UDTs.

```
Dim r1 As Rect
    Dim r2 As Rect
    :
    r1 = r2
```

When copying structures of the same type, all strings in the source UDT are duplicated and references are placed into the target UDT.

The LSet statement can be used to copy a UDT variable of one type to another.

```
LSet variable1 = variable2
```

LSet cannot be used with UDTs containing variable-length strings. The smaller of the two structures determines how many bytes get copied.

Passing Structures

UDTs can be passed both to user-defined routines and to external routines, and they can be assigned. UDTs are always passed by reference.

Since structures are always passed by reference, the ByVal keyword cannot be used when defining structure arguments passed to external routines (using Declare). The ByVal keyword can only be used with fundamental data types such as Integer and String.

Passing structures to external routines actually passes a far pointer to the data structure.

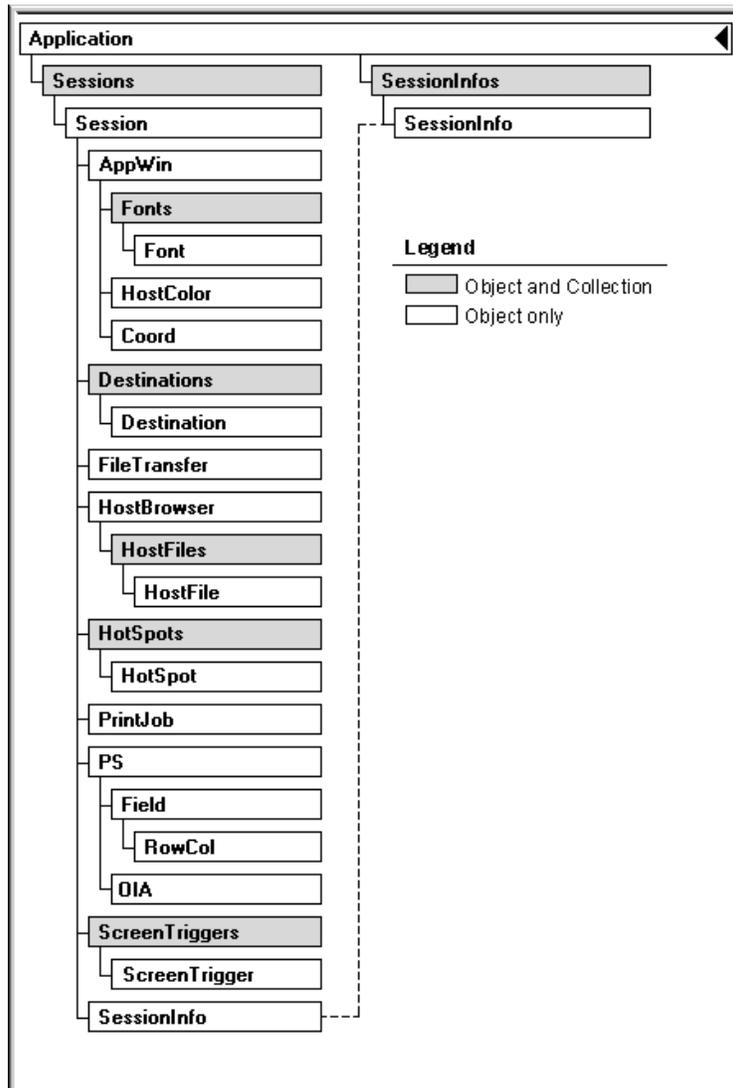
Size of Structures

The Len function can be used to determine the number of bytes occupied by a UDT.

`Len(udt_variable_name)`

Since strings are stored in AvivaBasic's data space, only a reference (currently, 2 bytes) is stored within a structure. Thus, the Len function may seem to return incorrect information for structures containing strings.

Application Object



This object is implemented in the Session Controller/Workspace Manager. It provides the workspace management functionality for all types of emulation sessions. This is the main object for the Session Controller with "Aviva.Application" as the Object ID.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined Application object name for Macro Scripts is "Application".

Workspace Management Methods

CloseAll	Closes all sessions opened in the workspace.
LastError	Retrieves the LastError value (ActiveX Automation only).
LastErrorMessage	Retrieves the error string that corresponds to the LastError value (Application and Session methods).
OpenWorkspaceFile	Opens the specified workspace file.
SaveWorkspaceAs	Saves the current workspace as a file and specifies a file name.

Collection Methods - (ActiveX Automation only)

Sessions	Returns an Automation Collection of Session objects. The number of session objects that you create is limited only by the available computing resources, (i.e. memory). Each session object has various methods and properties that manipulate and retrieve session information.
SessionInfos	Returns an Automation Collection of SessionInfo objects, one per active session. Each object in turn contains various methods to retrieve relevant session information.

Aviva Registry Path Methods

EiconConfigDir	Returns the Aviva configuration path.
EiconSystemDir	Returns the Aviva system path.
SessionDir	Returns the Aviva session path.
ExecutionDir	Returns the Aviva execution path.
MacroDir	Returns the Aviva macro path.
WorkspaceDir	Returns the Aviva workspace path.
LogFile	Returns the Aviva log file path and name.
TraceFile	Returns the Aviva trace file path and name.

CloseAll (method)**Object**

Application

Syntax**AvivaBasic Macro**`rc% = Application.CloseAll`**ActiveX controller application**`rc% = Object1.CloseAll`**Description**

CloseAll terminates all open sessions in the workspace.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
ERR_UNABLETOLOADDLL	Unable to load the required DLL.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_SESSIONNOTFOUND	Sessions are not available.
0	No errors were encountered.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

Remarks

Without exception, CloseAll terminates all open sessions in the workspace

See Also

[OpenWorkspaceFile \(method\)](#) on page 44

[SaveWorkspaceAs \(method\)](#) on page 46

[Close \(method\)](#) on page 234

[BlockClose \(method\)](#) on page 231

EiconConfigDir (method)

Object

Application

Syntax**AvivaBasic Macro**

```
ConfigDir$ = Application.EiconConfigDir
```

ActiveX controller application

```
ConfigDir$ = object1.EiconConfigDir
```

Description

Returns the Aviva configuration path.

Return Value

Returns *ConfigDir\$* as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[EiconSystemDir \(method\)](#) on page 40
[SessionDir \(method\)](#) on page 47
[ExecutionDir \(method\)](#) on page 40
[MacroDir \(method\)](#) on page 43
[WorkspaceDir \(method\)](#) on page 50
[LogFile \(method\)](#) on page 43
[TraceFile \(method\)](#) on page 49

EiconSystemDir (method)

Object

Application

Syntax

AvivaBasic Macro

```
SystemDir$ = Application.EiconSystemDir
```

ActiveX controller application

```
SystemDir$ = object1.EiconSystemDir
```

Description

Returns the Aviva system path.

Return Value

Returns *SystemDir\$* as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[EiconConfigDir \(method\)](#) on page 39
[SessionDir \(method\)](#) on page 47
[ExecutionDir \(method\)](#) on page 40
[MacroDir \(method\)](#) on page 43
[WorkspaceDir \(method\)](#) on page 50
[LogFile \(method\)](#) on page 43
[TraceFile \(method\)](#) on page 49

ExecutionDir (method)

Object

Application

Syntax

AvivaBasic Macro

```
ExeDir$ = Application.ExecutionDir
```

ActiveX controller application

```
ExeDir$ = object1.ExecutionDir
```

Description

Returns the Aviva execution path.

Return Value

Returns *ExeDir\$* as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[EiconConfigDir \(method\)](#) on page 39
[EiconSystemDir \(method\)](#) on page 40
[SessionDir \(method\)](#) on page 47
[MacroDir \(method\)](#) on page 43
[WorkspaceDir \(method\)](#) on page 50
[LogFile \(method\)](#) on page 43
[TraceFile \(method\)](#) on page 49

LastError (method)

Object

Application

Syntax

This method may only be executed from an ActiveX controller application.

```
MyLastError% = Object1.LastError
```

Description

Returns the last error encountered by the Application Interface.

Return Value

Returns *MyLastError%* as an integer.

Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.

To retrieve the string associated with the LastError value, refer to the **LastErrorMessage** method.

Prerequisite

Any command that cannot have the return code passing back the appropriate error number.

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

Remarks

Many Aviva ActiveX Automation and Macro language commands return error codes. However, some commands do not return error codes or they return a NULL string. In this case, LastError should be used to retrieve more information.

Once LastError has been called, its return value is reset to 0. It is up to the programmer to make the call at the correct time. For example, if LastError is executed at the wrong time then the error code returned may be for the previous command instead of the current one. Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.

LastErrorMessage (method)

Object

Application

Syntax

This method may only be executed from an ActiveX controller application.

```
MyLastError$ = Object1.LastErrorMessage(LastErrorNum)
```

Description

Retrieve the error string that corresponds to the value return by using the Application.**LastError** or Session.**LastError** methods.

Return Value

Returns *MyLastError\$* as a string.

Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.

Parameter

LastErrorNum is a valid error returned from the Application.LastError method.

Prerequisite

Any command that cannot have the return code passing back the appropriate error number.

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

Remarks

Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.

LogFile (method)

Object

Application

Syntax

AvivaBasic Macro

```
Logfile$ = Application.LogFile
```

ActiveX controller application

```
Logfile$ = object1.LogFile
```

Description

Returns the Aviva log file name with path.

Return Value

Returns Logfile\$ as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[EiconConfigDir \(method\)](#) on page 39

[EiconSystemDir \(method\)](#) on page 40

[SessionDir \(method\)](#) on page 47

[MacroDir \(method\)](#) on page 43

[ExecutionDir \(method\)](#) on page 40

[WorkspaceDir \(method\)](#) on page 50

[TraceFile \(method\)](#) on page 49

MacroDir (method)

Object

Application

Syntax

AvivaBasic Macro

```
MDir$ = Application.MacroDir
```

ActiveX controller application

```
MDir$ = object1.MacroDir
```

Description

Returns the Aviva macro path.

Return Value

Returns MDir\$ as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

- [EiconConfigDir \(method\)](#) on page 39
- [EiconSystemDir \(method\)](#) on page 40
- [ExecutionDir \(method\)](#) on page 40
- [SessionDir \(method\)](#) on page 47
- [WorkspaceDir \(method\)](#) on page 50
- [LogFile \(method\)](#) on page 43
- [TraceFile \(method\)](#) on page 49

OpenWorkspaceFile (method)

Object

Application

Syntax

AvivaBasic Macro

```
rc% = Application.OpenWorkSpaceFile(WorkSpaceFile$)
```

ActiveX controller application

```
rc% = Object1.OpenWorkSpaceFile(WorkSpaceFile$)
```

Description

Opens the specified workspace file.

Parameters

Element	Description
<i>WorkSpaceFile\$</i>	The name of the workspace file to be opened. The maximum length of the file, including the full path name, is 255 characters long.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
ERR_UNABLETOLOADDLL	Unable to load a DLL.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_INVALID_WORKSPACE_FILE	The specified workspace file is invalid.
ERR_CANNOT_START_SESSION	Cannot start a session
ERR_INVALID_SESSION_FILE	Invalid session file.
0	No errors were encountered.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

You can also open a workspace file by using the GetObject command. For example:

```
Dim AvivaApp as Object  
Set = AvivaApp GetObject("MyWorkspace.aws")
```

See Also

[SaveWorkspaceAs \(method\)](#) on page 46

[CloseAll \(method\)](#) on page 38

[Close \(method\)](#) on page 234

[Remove \(method\)](#) on page 273

SaveWorkspaceAs (method)

Object

Application

Syntax

AvivaBasic Macro

```
rc% = Application.SaveWorkSpaceAs (WorkSpaceFile$ [,Overwrite%])
```

ActiveX controller application

```
rc% = Object1.SaveWorkSpaceAs (WorkSpaceFile$ [,Overwrite%])
```

Description

Saves the workspace with the specified name.

Parameters	Description
<i>WorkSpaceFile\$</i>	The complete path and filename in which to save the workspace. The maximum length of the file, including the full path name, is 255 characters long.
<i>OverWrite%</i>	As Integer. Where <i>OverWrite%</i> can be set to constants TRUE or FALSE

Value	Description
FALSE(Default)	Don't overwrite file if it exists
TRUE	Overwrite file if it exists.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
ERR_UNABLETOLOADDLL	Unable to load the required DLL.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_INVALID_WORKSPACE_FILE	The specified workspace file is invalid.
ERR_INVALID_PARAM	An invalid parameter was specified.
ERR_FILEACCESSERROR	There was a file access error.
ERR_SESSIONNOTFOUND	There are no available sessions.
0	No errors were encountered.

Prerequisites

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[OpenWorkspaceFile \(method\)](#) on page 44
[CloseAll \(method\)](#) on page 38

SessionDir (method)

Object

Application

Syntax

AvivaBasic Macro

```
SessionDir$ = Application.SessionDir
```

ActiveX controller application

```
SessionDir$ = object1.SessionDir
```

Description

Returns the Aviva session path.

Return Value

Returns *SessionDir\$* as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[EiconConfigDir \(method\)](#) on page 39
[EiconSystemDir \(method\)](#) on page 40
[MacroDir \(method\)](#) on page 43
[ExecutionDir \(method\)](#) on page 40
[WorkspaceDir \(method\)](#) on page 50
[LogFile \(method\)](#) on page 43
[TraceFile \(method\)](#) on page 49

SessionInfos (method)

Object

Application

Syntax

This method may only be executed from an ActiveX controller application.

```
Set Object2 = Object1.SessionInfos([index%])
```

OR

```
Set Object2 = Object1.SessionInfos[SessionFileName$])
```

Description

Returns a collection of SessionInfo objects for all the open sessions, or a SessionInfo object of a single session.

Parameters	Description
<i>index%</i>	The session index number
<i>SessionFileName\$</i>	The session file name

Return Value(s)

Returns *Object2* as a SessionInfos object or a SessionInfo object.

Value	Meaning
Nothing	An error occurred.
LastError (method) - Session Object or LastError (Method) - Application Object should be executed to obtain the reason for the failure.	

Prerequisite

When executing from an ActiveX controller application, *Object1* must be an Aviva Application Object created using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

Remarks

This method returns an Automation Collection Object maintaining currently opened sessions if no parameters are passed to a SessionInfo object.

If *Index%* is used in the parameter, the Object Reference of the Session indexed by this value in the list of opened sessions is returned.

If *SessionFileName\$* is passed as the parameter, the Object Reference of the specified session (if opened) is returned.

See Also

[SessionInfo Object](#) on page 275
[SessionInfo \(property\)](#) on page 256
[SessionInfos Object](#) on page 284

Sessions (method)

Object

Application

Syntax

This method may only be executed from an ActiveX controller application.

```
Set MySessionObj = object1.Sessions ([index$]  
OR  
Set MySessionObj = object1.Sessions [SessionFileName$])
```

Description

Returns a reference to a collection of session objects.

Parameters	Description
<i>index%</i>	The session index number
<i>SessionFileName\$</i>	The session file name.

Return Values

Returns *MySessionObj* as an object.

Value	Meaning
Nothing	An error occurred.
LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.	
Last Error Value	Meaning
ERR_INVALID_PARAM	Invalid parameter, verify syntax.
ERR_INVALID_SESSION_FILE	Invalid session file.
ERR_SESSIONNOTFOUND	There are no available sessions
ERR_OLENOTAUTOMATIONOBJECT	Could not activate ActiveX Automation.
ERR_FAIL	A system error occurred.
ERR_OUTOFMEMORY	Not enough memory to create object.

Prerequisite

When executing from an ActiveX controller application, *Object1* must be an Aviva Application Object created using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

Remarks

If Sessions is called without any parameters, the Collection Object Reference is returned. The user can then start using the available methods.

If *Index%* is specified in the parameters, the Object Reference of the Session indexed by this value in the list of opened sessions is returned.

If *SessionFileName\$* is specified in the parameters, the Object Reference of the specified session (if opened) is returned.

See Also

[Session Object](#) on page 228

[Sessions Object](#) on page 267

TraceFile (method)

Object

Application

Syntax

AvivaBasic Macro

```
TraceDir$ = Application.TraceFile
```

ActiveX controller application

```
TraceDir$ = object1.TraceFile
```

Description

Returns the Aviva trace file name with path.

Return Value

Returns *TraceDir\$* as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[EiconConfigDir \(method\)](#) on page 39

[EiconSystemDir \(method\)](#) on page 40

[SessionDir \(method\)](#) on page 47

[MacroDir \(method\)](#) on page 43

[ExecutionDir \(method\)](#) on page 40

[WorkspaceDir \(method\)](#) on page 50

[LogFile \(method\)](#) on page 43

WorkspaceDir (method)

Object

Application

Syntax

AvivaBasic Macro

```
WorkspaceDir$ = Application.WorkspaceDir
```

ActiveX controller application

```
WorkspaceDir$ = object1.WorkspaceDir
```

Description

Returns the Aviva workspace path.

Return Value

Returns *WorkspaceDir\$* as a string.

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

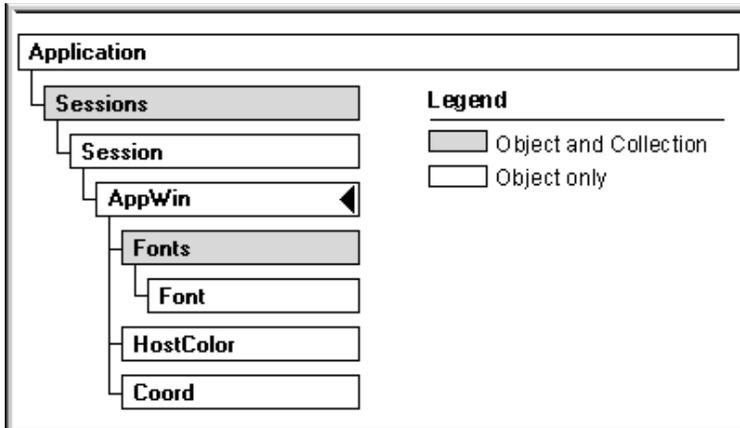
```
Set AvivaApp = CreateObject("Aviva.Application")
```

See Also

[EiconConfigDir \(method\)](#) on page 39
[EiconSystemDir \(method\)](#) on page 40
[SessionDir \(method\)](#) on page 47
[MacroDir \(method\)](#) on page 43
[ExecutionDir \(method\)](#) on page 40
[LogFile \(method\)](#) on page 43
[TraceFile \(method\)](#) on page 49

AppWin Object

Display sessions only



This object provides processing for the graphic user interface (GUI). Macro scripts and client applications can obtain the AppWin object reference from the Session object - Appwin property.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document. You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for macro scripts is "AppWin".

Methods and properties of the AppWin object are as follows:

Edit Menu	
Copy	Copies screen data to the clipboard.
GetPosition	Returns the window's position (row,column).
Paste	Pastes clipboard data to the screen.
SaveScreen	Saves the screen image to a specified file.
PrintScreen	Prints the screen image to a specified printer.
Select	Selects text in the application window.
SelectAll	Selects all of the text in the application window.
Deselect	Reverses the most recent Select command.
SetPosition	Sets the window's position.
SetSize	Sets the window's width and height.
Fonts Manipulation	
IncreaseFontSize	Increments the font size of the host display area.
DecreaseFontSize	Decrements the font size of the host display area.
Object Creation	

GetSize	Gets the window's width and height and return a Coord object
Fonts	Returns a Fonts collection object.
HostColor (3270 and 5250 only)	Retrieves the Host Color object that is maintained by AppWin.
GUI	
ShowState	Gets or sets the show state of the application window.
Visible	Shows or hides the application window.
DisplayMode (3270 and 5250 only)	Retrieves or sets the display mode of a session

See Also

[AppWin \(property\)](#) on page 231

Copy (method)

Display sessions only**Object**

AppWin

Syntax**AvivaBasic Macro**

rc% = `AppWin.Copy`

ActiveX client application

rc% = `Object1.Copy`

Description

Copies screen data to the clipboard. When you paste this data into your application, you can correct any display problems by selecting a Courier font.

You can record this method by using Aviva's macro recorder. For a list of AvivaBasic language items that you can record, see Recording AvivaBasic Macros macros topic.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
```

DecreaseFontSize (method)**Display sessions only****Object**

AppWin

Syntax**AvivaBasic Macro**

```
rc% = AppWin.DecreaseFontSize
```

ActiveX client application

```
rc% = Object1.DecreaseFontSize
```

Description

This method decrements the font size of the host display.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
```

See Also

[IncreaseFontSize \(method\)](#) on page 60

Deselect (method)

Display sessions only

Object

AppWin

Syntax

AvivaBasic Macro

```
rc% = AppWin.Deselect
```

ActiveX client application

```
rc% = Object1.Deselect
```

Description

Reverses the most recent Select command.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObjObj = MySession.AppWin
```

DisplayMode (property)

3270 and 5250 display sessions only

Object

AppWin

Syntax

AvivaBasic Macro

```
DisplayMode% = AppWin.DisplayMode  
AppWin.DisplayMode = DisplayMode%
```

ActiveX client application

```
DisplayMode% = Object1.DisplayMode
Object1.DisplayMode = DisplayMode%
```

Note You can set more than one display mode by doing the following:

Set to 3D display and OIA as text

```
DisplayMode% = ebxDispMode3D OR ebxDispModeOiaAsText
```

Description

Retrieve or set the display mode of a session. See **About 3D mode** in the Aviva Help Topics for a description of Aviva 3D display mode and limitations.

Return Value(s)

Returns *DisplayMode%* as an integer.

Constants	Value	Description
ebxDispModeNormal	0	None of the modes are set
ebxDispMode3D	1	Sets a session to a Windows-style dialog box (3D)
ebxDispModeOiaAsText	2	Hides the OIA display. OIA information appears as text on the status bar

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
```

Fonts (method)

Display sessions only**Object**

AppWin

Syntax

In AvivaBasic, **Fonts** is a predefined object.

In AvivaBasic to access a Font object, use the Font data type therefore,

```
Dim Object2 as Font
Set Object2 = Fonts([Index% OR FontName$])
```

ActiveX client application

```
Set Object2 = Object1.Fonts([Index% OR FontName$])
```

Description

Returns an object reference for a Fonts Automation Collection, or an object reference for a specified Font.

Parameters

Element	Description
<i>index%</i>	The font index number in the list of Fonts.
<i>FontName\$</i>	The font name in the list of Fonts.

Return Value(s)

Returns MyFontObj as a Fonts Object.

Value	Meaning
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
```

See Also

[Font Object](#) on page 114
[Fonts Object](#) on page 118

GetPosition (method)

Display sessions only**Object**

AppWin

Syntax**AvivaBasic Macro**

```
Set CoordObj = AppWin.GetPosition
(where Object1 is Dim Object1 as Coord)
```

ActiveX client application

```
Set CoordObj = Object1.GetPosition
```

Description

Retrieves the upper left corner position of the application window and returns it as a Coord object.

Return Value(s)

Returns *CoordObj* as a Coord Object.

Value	Description
CoordObj	As Coord (for macro). As Object (for ActiveX Automation).
Nothing	Command failure.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
AvivaBasic:
Dim CoordObj as Coord
```

GetSize (method)

Display sessions only**Object**

AppWin

Syntax**AvivaBasic Macro**

```
Set Object1 = AppWin.GetSize
(where Object1 is Dim Object1 as Coord)
```

ActiveX client application

```
Set MyCoordObj = Object1.GetSize
```

Description

Retrieves the application window's width and height, and returns them in the Coord object.

Return Value(s)

Returns *CoordObj* as a Coordinate Object.

Value	Description
Object	As Coord Object(for macro). As Object (for ActiveX Automation).
Nothing	Command failure.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
```

AvivaBasic:

```
Dim MyCoordObj as Coord
```

HostColor (property)

3270 and 5250 display sessions only**Object**

AppWin

Syntax

In AvivaBasic, **Hostcolor** is a predefined object.

The following syntax applies only to an ActiveX client application.

```
Set Object2 = Object1.HostColor
```

Description

Return a HostColor object.

Return Value(s)

Returns *Object2* as an object.

Value	Meaning
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set MyHostColorObj = AppWinObj.Hostcolor
```

See Also

[HostColor Object](#) on page 130

IncreaseFontSize (method)

Display sessions only

Object

AppWin

Syntax

AvivaBasic Macro

```
rc% = AppWin.IncreaseFontSize
```

ActiveX client application

```
rc% = Object1.IncreaseFontSize
```

Description

This method increments the font size of the host display.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin
```

See Also

[DecreaseFontSize \(method\)](#) on page 54

Paste (method)

Display sessions only

Object

AppWin

Syntax

AvivaBasic Macro

```
rc% = AppWin.Paste
```

ActiveX client application

```
rc% = Object1.Paste
```

Description

Paste clipboard data to screen. When you paste this data into your application, you can correct any display problems by selecting a Courier font.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
```

PrintScreen (method)

Display sessions only**Object**

AppWin

Syntax**AvivaBasic Macro**

```
rc% = AppWin.PrintScreen(PrinterName$)
```

ActiveX client application

```
rc% = Object1.PrintScreen(PrinterName$)
```

Description

Print the screen image to a specified printer.

Parameters

Parameter	Description
<i>PrinterName\$</i>	The printer name. Maximum length of the printer name is 255 characters. Use "" for the current default printer.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	Successful
<i>ERR_SYSTEM_ERROR</i>	A system error was encountered.
<i>ERR_NO_PRINTER</i>	The specified printer name does not exist.
<i>ERR_FAIL</i>	Session terminating (ActiveX client only).
Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.	

SaveScreen (method)

Display sessions only**Object**

AppWin

Syntax**AvivaBasic Macro**

```
rc% = AppWin.SaveScreen(ScreenFile$ [, flag%])
```

ActiveX client application

```
rc% = Object1.SaveScreen(ScreenFile$ [, flag%])
```

Description

Save the screen image to a specified file.

Parameters	Description
<i>ScreenFile\$</i>	The screen image file name. Maximum length of the file name is 255 characters.
<i>Flag%</i>	As integer.

Constants	Flag% Value	Meaning
ebxSaveScreenDefault	0 (default)	Opens a new file. However, if the file exists then overwrite. This action may not overwrite the entire file.
ebxSaveScreenOverwrite	1	If this file exists, overwrite the entire file.
ebxSaveScreenAppend	2	If this file exists, append to the file.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	Successful
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_INVALID_PARAM	Invalid parameter specified.
ERR_FILEALREADYEXISTS	File already exists.
ERR_FILEACCESSERROR	There was an error accessing the file.
ERR_FAIL	Session terminating (ActiveX client only).
Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.	

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
```

Select (method)**Display sessions only****Object**

AppWin

Syntax**AvivaBasic Macro**

```
rc% = AppWin.Select(top%, left%, bottom%, right%)
```

ActiveX client application

```
rc% = Object1.Select(top%, left%, bottom%, right%)
```

Description

Selects text in the application window. You can then copy, paste, or save text by using other Appwin methods.

Parameters

Value	Description
top%	An integer that specifies the top edge of the screen to be selected.
left%	An integer that specifies the left edge of the area to be selected.
bottom%	An integer that specifies the bottom edge of the screen to be selected.
right%	An integer that specifies the right edge of the area to be selected.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
```

SelectAll (method)

Display sessions only**Object**

AppWin

Syntax**AvivaBasic Macro**

```
rc% = AppWin.SelectAll
```

ActiveX client application

```
rc% = Object1.SelectAll
```

Description

Selects all the text in the application window. You can then copy, paste, or save text by using other Appwin methods.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
```

SetPosition (method)

Display sessions only

Object

AppWin

Syntax

AvivaBasic Macro

```
rc%=AppWin.SetPosition(left%, top%)
```

ActiveX client application

```
rc%=Object1.SetPosition(left%, top%)
```

Description

Set the upper left corner position of the application window using the specified parameters.

Parameters

Element	Description
<i>left%</i>	Integer specifying the X coordinate in pixels of the upper left corner position of the application window.
<i>top%</i>	Integer specifying the Y coordinate in pixels of the upper left corner position of the application window.

Return Value(s)

rc% as Integer.

Value	Meaning
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObjObj = MySession.AppWin
```

SetSize (method)

Display sessions only**Object**

AppWin

Syntax**AvivaBasic Macro**

```
rc%=AppWin.SetSize(width%, height%)
```

ActiveX client application

```
rc%=Object1.SetSize(width%, height%)
```

Description

Set the width and height of the window.

Parameters

Element	Description
<i>width%</i>	An integer specifying the total width in pixels of the application window.
<i>height%</i>	An integer specifying the total height in pixels of the application window.

Return Value(s)

rc% as Integer.

Value	Meaning
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
```

ShowState (property)

Display sessions only

Object

AppWin

Syntax

AvivaBasic Macro

```
State% = AppWin.ShowState  
AppWin.ShowState = State%
```

ActiveX client application

```
State% = Object1.ShowState  
Object1.ShowState = State%
```

Description

Retrieve or set the state of the current application window.

Return Value(s)

Returns *State%* as integer.

Constants	Value	Description
ebxShowNormal	1	Activates and displays a window whether the window is minimized or maximized. The window is restored to its original size and position. An application should specify this flag when displaying a window for the first time
ebxShowMinimized	2	Minimize the application window.
ebxShowMaximized	3	Maximize the application window.
ebxMinimize	4	Minimizes the specified window and activates the next top-level window in the Z order
ebxShowMinNoActive	5	Displays the window as a minimized window. The active window remains active.
ebxShowNoActivate	6	Displays a window in its most recent size and position. The active window remains active.
ebxRestore	7	Restore the application window.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin
```

Visible (property)

Display sessions only

Object

AppWin

Syntax

AvivaBasic Macro

```
Bool = AppWin.Visible  
AppWin.Visible = Bool
```

ActiveX client application

```
Bool = Object1.Visible  
Object1.Visible = Bool
```

Description

Sets the current session window as visible, or invisible.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	The session window is not visible.
TRUE	The session window is visible.

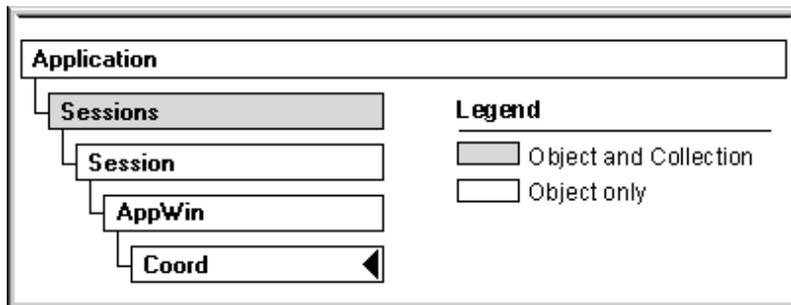
Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin
```

Coordinate Object

Display sessions only



Window position manipulation properties

This object contains the x and y coordinates or size of an Aviva display session window. You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined data type for Macro Scripts is "Coord".

Properties

x	Returns the x coordinate.
y	Returns the y coordinate.

Remark

In AvivaBasic, Coord is a data type therefore, Dim MyObject as Coord.

See Also

[GetSize \(method\)](#) on page 58

X (property)

Display sessions only

Object

Coordinate Object

Syntax

AvivaBasic

`X% = Object1.X`

(where *Object1* is Dim *Object1* as Coord)

ActiveX controller application

`X% = Object1.X`

Description

Returns the X coordinate from a coordinate object created by executing **GetSize** or **GetPosition**.

Return Value(s)

Returns X% as Integer.

Prerequisite

When executing from an ActiveX controller application, *Object1* must be a Coord object created using one of the following methods:

```
Set MySession = GetObject("MySession.a3d")  
Set MyCoordObj = MySession.AppWin.getsize
```

See Also

- [X \(property\)](#) on page 69
- [Row \(property\)](#) on page 217
- [Column \(property\)](#) on page 217

Y (property)

Display sessions only

Object

Coordinate Object

Syntax

AvivaBasic

`Y% = Object1.Y`
(where *Object1* is Dim *Object1* as Coord)

ActiveX controller application

`Y% = Object1.Y`

Description

Returns the Y coordinate from a coordinate object created by executing **GetSize** or **GetPosition**.

Return Value(s)

Returns Y% as Integer.

Prerequisite

When executing from an ActiveX controller application, *Object1* must be a Coord object created using one of the following methods:

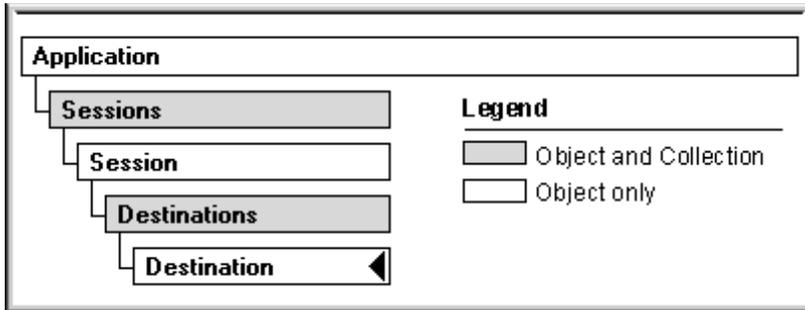
```
Set MySession = GetObject("MySession.a3d")  
Set MyCoordObj = MySession.AppWin.getsize
```

See Also

- [X \(property\)](#) on page 69
- [Row \(property\)](#) on page 217
- [Column \(property\)](#) on page 217

Destination Object

TN3270 display sessions only



This object includes properties to retrieve information about a specific destination including its connection type and name, and connection information such as the server, LU, PU, and port number that the destination uses.

Properties

ConnectionTypes	Returns the connection type of the destination.
DestinationName	Returns the name of the destination.
DeviceName	Returns the name of the device used by the session.
HostName	Returns the name or IP address of the server used by the destination.
LuNumber	Returns the LU number used by the destination.
PortNumber	Returns the port number used by the destination.
PuName	Returns the name of the PU used by the destination.
Security	Returns whether the destination uses encryption.
ServerName	Returns the name of the server used by the destination.

DestGeneric Interface

TN3270 display sessions only

To obtain common properties for all connection types, use the DestGeneric interface. DestGeneric can obtain the following properties:

- [ConnectionType \(property\)](#) on page 73
- [DestinationName \(property\)](#) on page 74

Example

```
Dim MySession As Object
Dim MyDest As DestGeneric
Set MySession = GetObject("MySession.a3d")
Set MyDest = MySession.Destination(1)
MsgBox MyDest.DestinationName
```

Dest32TN Interface

TN3270 display sessions only

To obtain properties for the TN3270 connection type only, use the Dest32TN interface.

Dest32TN can obtain the following properties:

[DeviceName \(property\)](#) on page 74

[HostName \(property\)](#) on page 75

[PortNumber \(property\)](#) on page 75

[Security \(property\)](#) on page 75

Example

```
Dim MySession As Object
Dim MyDest As Dest32TN
Set MySession = GetObject("MySession.a3d")
Set MyDest = MySession.Destination(1)
MsgBox MyDest.HostName
```

ConnectionType (property)

TN3270 display sessions only**Object**

Destination

Syntax

This property can be executed only from an ActiveX client application.

object.ConnectionType

Description

Returns the connection type of the destination.

Return Value(s)

Returns *object* as Long.

Value	Description
1	Trace Player
3	Microsoft Host Integration Server
11	IBM® Communications Server
12	TN3270/TN5250/Telnet VT

DestinationName (property)

TN3270 display sessions only

Object

Destination

Syntax

This method can be executed only from an ActiveX client application.

object.DestinationName

Description

Returns the name of the destination.

Return Value(s)

Returns *object* as a string.

DeviceName (property)

TN3270 display sessions only

Object

Destination

Syntax

This property can be executed only from an ActiveX client application.

object.DeviceName

Description

Returns the name of the device used by the destination.

Return Value(s)

Returns *object* as a string.

HostName (property)

TN3270 display sessions only

Object

Destination

Syntax

This property can be executed only from an ActiveX client application.

object.HostName

Description

Returns the name or the IP address of the server used by the destination.

Return Value(s)

Returns *object* as string.

PortNumber (property)

TN3270 display sessions only

Object

Destination

Syntax

This method can be executed only from an ActiveX client application.

object.PortNumber

Description

Returns the port number used by the destination.

Return Value(s)

Returns *object* as Long.

Security (property)

TN3270 display sessions only

Object

Destination

Syntax

This property can be executed only from an ActiveX client application.

Bool = *object.Security*

Description

Returns whether the destination uses encryption.

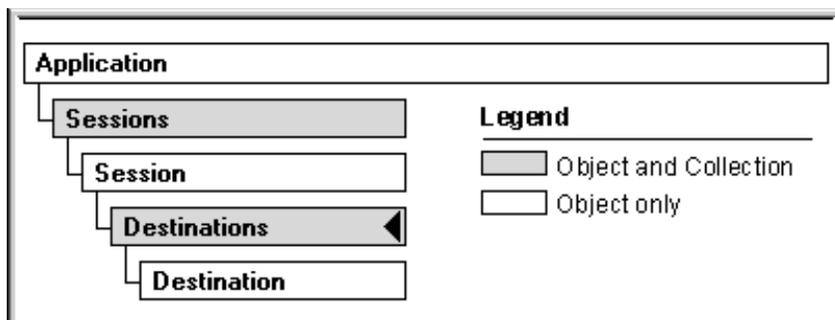
Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	Destination does not use encryption.
TRUE	Destination uses encryption.

Destinations Object

TN3270 display sessions only



This object provides access to all the destinations loaded in a session. It includes methods and properties to return the number of destinations, to access a specific destination from the destination list, and to retrieve the destination that is being used by the session.

Methods

Item	Returns a specific destination from the list of destinations configured for a session.
-------------	--

Properties

Count	Returns the number of destinations in the collection
DestinationInUse	Returns the destination that is currently in use to connect the session. If the session is not connected to the host, this method returns a null object.

Count (property)

TN3270 display sessions only

Object

Destinations

Syntax

This method can be executed only from an ActiveX client application.

object.count

Description

Returns the number of destinations in the collection.

Return Value(s)

Returns *object* as Long.

Item (method)

TN3270 display sessions only

Object

Destinations

Syntax

This method can be executed only from an ActiveX client application.

```
Set MyDest = object.Item(Index&)
```

Description

Returns a specific destination from the list of destinations configured for a session.

Parameters	Description
Index	Index of the destination in the collection as Long.

Return Value(s)

Returns MyDest as Object.

Remarks

Returns the Destination object at the specified index (start from 1).

DestinationInUse (property)

TN3270 display sessions only

Object

Destinations

Syntax

This method can be executed only from an ActiveX client application.

```
Set MyDest = object.DestinationInUse
```

Description

Returns the destination that is currently in use to connect the session. If the session is not connected to the host, the method returns a null object.

Return Value(s)

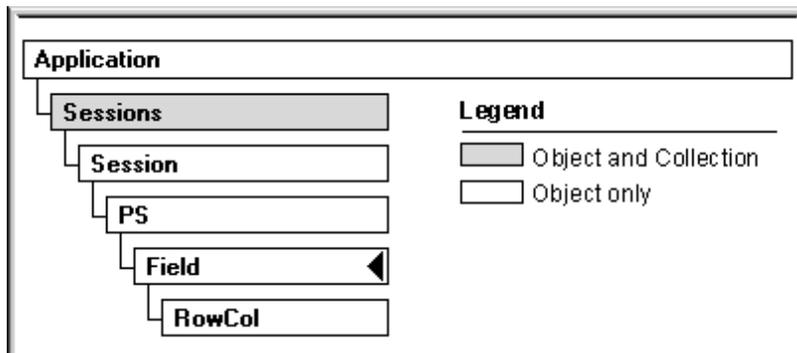
Returns *MyDest* as an object.

Remarks

When the connection state changes, the Destinations collection must be recreated before querying for the destination in use.

Field Object

3270 and 5250 display sessions only



This object provides processing for handling fields on the host presentation space. Macro scripts and client applications can obtain the Field object reference from the PS object - Field method.

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation		
Model	Rows	Columns	Model	Rows	Columns
2	24	80	Normal	24	80
3	32	80	Alternate	27	132
4	43	80			
5	27	132			
11	62	160			

The 5250 emulation type supports Model 2 and Model 5 screens. 5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined data type for Macro Scripts is "Field".

Field Manipulation Methods

GetData	Gets the contents of a specified field in the presentation space.
SetData	Copies a string into the specified field in the presentation space.
Next	Returns the next field object.
NextProtected	Returns the next protected field object.
NextUnProtected	Returns the next unprotected field object.
Position	Gets the position (row,column) of the field.
Prev	Returns the previous field object.
PrevProtected	Returns the previous protected field object.
PrevUnProtected	Returns the previous unprotected field object.

Properties

Attribute	Returns the attribute byte of the field.
IsProtected	Specifies if the current field type is protected (or unprotected).
Length	Returns the length of the field.

Remarks

In AvivaBasic, Field is a data type, therefore, `Dim MyObject as Field`.

If you declare the Field Object as `Dim MyObject as Field`, you must not destroy this object by using the Set statement `Set MyObject = Nothing`. When your script completes, the Aviva Macro Editor will manage this object.

See Also

[Field \(method\)](#) on page 184

Attribute (property)**3270 and 5250 display sessions only****Object**

Field

Syntax**AvivaBasic Macro**

```
Attribute% = PS.Field (x,y).Attribute
```

ActiveX client application

```
Attribute% = Object1.Attribute
```

Description

Returns the attribute byte of a field.

Return Value(s)

Returns *Attribute%* as an integer.

Prerequisite

Before executing *Attribute* on a presentation space your script or program must set **SetSharing** to *TO_QUERY* on the Session object from which PS was created. It is not necessary to repeat this action.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyFieldObject = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

Remarks

The attribute byte returned is a value equal to, or greater than hex value C0. Refer to the WinHLLAPI specification on QueryFieldAttribute (function 14) for the meaning of the value returned.

WHLLAPI Reference

WHLLAPI QueryFieldAttribute (function 14)

GetData (method)

3270 and 5250 display sessions only**Object**

Field

Syntax**AvivaBasic Macro**

```
fieldData$ = PS.Field (x,y).GetData(length%)
```

ActiveX client application

```
fieldData$ = Object1.GetData(length%)
```

Description

Copy the contents of the specified field from the presentation space.

What is copied from the presentation space depends on the values set by the following PS properties, **Attrib**, **ExtendedAttrib**, and **NullToSpace**.

Parameters

Element	Description
<i>length%</i>	The length of the data string.

Return Value(s)

Returns *fieldData\$* as a string:

Value	Description
Any string	The requested portion of the session's presentation space, in the form of a string.
" "	Failed to retrieve the requested portion of the session's presentation space. Use the LastError() (LastError - Session Object (method)) method to obtain the reason for the failure.
LastError value	Description
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
ERR_SYSTEM_ERROR	System error.

Prerequisites

Before executing GetData on a presentation space your script or program must set **SetSharing** to TO_READ on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFieldObj = PS.Field(x,y)
```

WHLLAPI Reference

WHLLAPI CopyFieldToString (function 34)

See Also

[SetSharing \(method\)](#) on page 260

IsProtected (property)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Bool = PS.Field (x,y).IsProtected
```

ActiveX client application

`Bool = Object1.IsProtected`

Description

Indicates if the field within the specified row and column is protected or not.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	Field is unprotected.
TRUE	Field is protected.

Prerequisite

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFieldObject = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

WHLLAPI Reference

WHLLAPI QueryFieldAttribute (function 14)

Length (property)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
length% = PS.Field (x,y).Length
```

ActiveX client application

```
length% = Object1.Length
```

Description

Returns the length of the current field.

Return Value(s)

Returns *Length%* as an integer.

Prerequisite

Before executing Length on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFieldObject = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

WHLLAPI Reference

WHLLAPI FindFieldLength (function 32)

Next (method)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Set MyFieldObj = PS.Field (x,y).Next
```

ActiveX client application

```
Set FieldObj1 = Object1.Next
```

Description

The field that follows the current field in the presentation space is returned as a Field object.

Return Value(s)

Returns *MyFieldObj* as a Field Object.

Value	Description
MyFieldObj	As Field (for macro).
	As Object (for ActiveX Automation).
<i>Nothing</i>	No field to return or session is terminating (ActiveX client only).

Prerequisite

Before executing Next on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFieldObj = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

See Also

[SetSharing \(method\)](#) on page 260

NextProtected (method)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Set MyFieldObj = PS.Field (x,y).NextProtected
```

ActiveX client application

```
Set MyFieldObj = Object1.NextProtected
```

Description

The first protected field that follows the current field in the presentation space is returned as a Field object.

Return Value(s)

Returns *MyFieldObj* as a Field Object.

Value	Description
MyFieldObj	As Field (for macro).
	As Object (for ActiveX Automation).
<i>Field</i>	The next protected field.
<i>Nothing</i>	No field to return or session is terminating (ActiveX client only).

Prerequisite

Before executing NextProtected on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFieldObj = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

See Also

[SetSharing \(method\)](#) on page 260

NextUnprotected (method)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Set MyFieldObj = PS.Field (x,y).NextUnprotected
```

ActiveX client application

```
Set MyFieldObj = Object1.NextUnprotected
```

Description

The first next unprotected field that follows the current field in the presentation space is returned as a Field object.

Return Value(s)

Returns *MyFieldObj* as a Field Object.

Value	Description
MyFieldObj	As Field (for macro).
	As Object (for ActiveX Automation).
<i>Field</i>	The next unprotected field.
<i>Nothing</i>	No field to return, or session is terminating (ActiveX client only).

Prerequisite

Before executing NextUnprotected on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = SetObject("MySession.a3d")
Set MyFieldObj = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

See Also

[SetSharing \(method\)](#) on page 260

Position (property)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Set RowColObj = PS.Field (x,y).Position
(where RowColObj is Dim RowColObj as RowCol)
```

ActiveX client application

```
Set Object2 = Object1.Position
```

Description

Returns an object reference for the starting position of a specified field.

Return Value(s)

RowColObj as a RowCol Object.

Value	Description
RowColObj	As RowCol (for macro).
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisites

Before executing Position on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFieldObj = MySession.PS.Field(1, 19) 'Must be valid
positions
```

AvivaBasic Macro

```
Dim RowColObj as RowCol
```

See Also

[RowCol Object](#) on page 216
[SetSharing \(method\)](#) on page 260

Prev (method)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Set MyFieldObj = PS.Field (x,y).Prev
```

ActiveX client application

```
Set MyFieldObj = Object1.Prev
```

Description

The field that is before the current field in the presentation space is returned as a Field object.

Return Value(s)

Returns *MyFieldObj* as a Field Object.

Value	Description
MyFieldObj	As Field (for macro).
	As Object (for ActiveX Automation).
<i>Field</i>	The previous field.
<i>Nothing</i>	No field to return or session is terminating (ActiveX client only).

Prerequisite

Before executing `Prev` on a presentation space your script or program must set **SetSharing** to `TO_QUERY` on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyFieldObj = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

See Also

[SetSharing \(method\)](#) on page 260

PrevProtected (method)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Set MyFieldObj = PS.Field (x,y).PrevProtected
```

ActiveX client application

```
Set MyFieldObj = Object1.PrevProtected
```

Description

The first protected field before the current field in the presentation space is returned as a Field object.

Return Value(s)

Returns *MyFieldObj* as a Field Object.

Value	Description
MyFieldObj	As Field (for macro).
	As Object (for ActiveX Automation).
<i>Field</i>	The previous protected field.
<i>Nothing</i>	No field to return or session is terminating (ActiveX client only).

Prerequisite

Before executing PrevProtected on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFieldObj = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

See Also

[SetSharing \(method\)](#) on page 260

PrevUnProtected (method)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
Set MyFieldObj = PS.Field (x,y).PrevUnProtected
```

ActiveX client application

```
Set MyFieldObj = Object1.PrevUnProtected
```

Description

The first unprotected field before the current field in the presentation space is returned as a Field object.

Return Value(s)

Returns *MyFieldObj* as a Field Object.

Value	Description
MyFieldObj	As Field (for macro).
	As Object (for ActiveX Automation).
<i>Field</i>	The previous unprotected field.
<i>Nothing</i>	No field to return or session is terminating (ActiveX client only).

Prerequisite

Before executing PrevUnprotected on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyFieldObject = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

See Also

[SetSharing \(method\)](#) on page 260

SetData (method)

3270 and 5250 display sessions only

Object

Field

Syntax

AvivaBasic Macro

```
rc% = PS.Field (x,y).SetData(string$)
```

ActiveX client application

```
rc% = Object1.SetData(string$)
```

Description

Copies a string to the specified field in the host presentation space.

Parameters

Element	Description
<i>string\$</i>	The string to send to the desired field in the host presentation space.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The string was successfully copied to the host.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
ERR_SESSION_INHIBITED	Target field is protected or inhibited, or illegal data sent to the target field.
ERR_SYSTEM_ERROR	System error.

Prerequisite

Before executing SetData on a presentation space your script or program must set **SetSharing** to SHARE_WITH_ALL on the Session object from which PS was created. If you have already done this it is not necessary to repeat it.

From an ActiveX client application, create a Field object by using the Field method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyFieldObj = PS.Field(x,y)
```

AvivaBasic Macro

```
Dim MyFieldObj as Field
```

WHLLAPI Reference

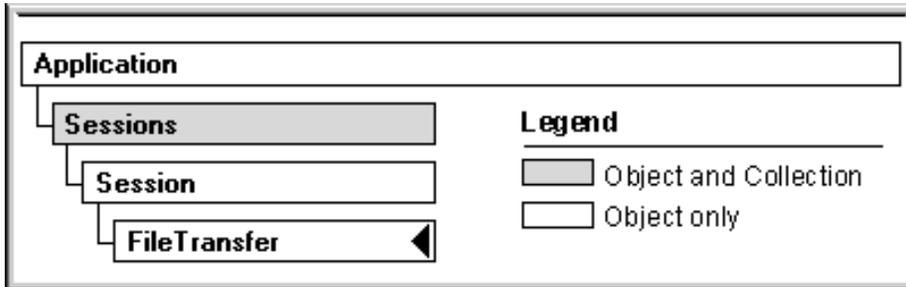
WHLLAPI CopyStringToField (function 33)

See Also

[SetSharing \(method\)](#) on page 260

FileTransfer Object

3270 display sessions only



Aviva transfers files to and from host systems using the IND\$FILE™ protocol. You can send a file to the host using data storage options available for many environments including CICS, TSO, VM/CMS and Proginet™ (IND\$FILE+). In addition to data storage options, you can set other parameters before you transfer a file. The host must be properly configured before you can use FileTransfer.

Macro scripts and client applications can obtain the FileTransfer object reference from the Session object - FileTransfer property.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for macro scripts is "FileTransfer".

Asynchronous File Transfer

The user application has to monitor the transfer by inquiring the properties Status, or BytesTransferred before proceeding with the next program statement.

Synchronous File Transfer

FileTransfer does not return until the transfer is complete, or an error has occurred.

File Transfer Methods

Send	Sends a file to the host.
Receive	Receives a file from the host.
Abort	Stops a file transfer.
BytesTransferred	Returns the number of bytes sent or received.
Status	Retrieves the status of the file transfer.
Reset	Resets all file transfer properties to default values.

File Transfer Properties

Append	Appends data to the existing file.
BlockSize	Sets or retrieves the size of data blocks in a new data set on the MVS/TSO volume.
CrLf	Controls the carriage return and line feed code.
HostFile	Sets or retrieves the host file name
LogRecLen	Sets or retrieves the file record length.
PacketSize	Sets or retrieves the size (in bytes) of a packet.
PCFile	Sets or retrieves the file name.
RecFormat	Sets or retrieves the record format and characteristics of host file.
SpaceAlloc	Retrieves or selects the type of space allocation.
SpaceIncrement	Retrieves or allocates the amount of secondary space.
SpaceQuantity	Retrieves or allocates the amount of primary space.
TimeOut	Sets the length of time (in seconds) that a program waits for a response from the host before sending an error message.
Translate	Sets ASCII to EBCDIC translation.
UserOptions	Specifies additional parameters for the file transfer.
Scheme	Specifies a file transfer scheme.

See Also

[FileTransfer \(property\)](#) on page 241

Abort (method)**3270 display sessions only****Object**

FileTransfer

Syntax**AvivaBasic Macro**

`FileTransfer.Abort`

ActiveX client application

`Object1.Abort`

Description

Stops the current file transfer. If there is no file transfer in progress, this command does nothing.

Prerequisite

From an ActiveX client application, FileXfer must be a FileTransfer Object created using the FileTransfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

See Also

[Send \(method\)](#) on page 106
[Receive \(method\)](#) on page 102
[BytesTransferred \(method\)](#) on page 96
[Status \(method\)](#) on page 110
[Reset \(method\)](#) on page 104

Append (property)

3270 display sessions only**Object**

FileTransfer

Syntax**AvivaBasic Macro**

```
rc% = FileTransfer.Append
FileTransfer.Append = rc%
```

ActiveX client application

```
rc% = Object1.Append
```

Description

The file transferred is appended to an existing file.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Constants	Value	Description
ebxNotAppend	0	Do not append.
ebxAppend	1	Append.
ebxDefault	2	Use the host default setting

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

BlockSize (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc& = FileTransfer.BlockSize
FileTransfer.BlockSize = rc&
```

ActiveX client application

```
rc& = Object1.BlockSize
Object1.BlockSize = rc&
```

Description

Sets the size of the data blocks in a new data set on the MVS/TSO volume.

Return Value(s)

Returns *rc&* as Long.

Value	Description
<i>rc&</i>	Size of the data block, from 1 to 65536 bytes. The default value is 1.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

BytesTransferred (method)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc& = FileTransfer.BytesTransferred
```

ActiveX client application

```
rc& = Object1.BytesTransferred
```

Description

Gets the number of bytes sent.

Return Value(s)

Returns *rc&* (long value) as the number of bytes transferred in the current file transfer operation.

Value	Description
<i>rc&</i>	The number of bytes transferred.
ERR_FAIL	Session is terminating (ActiveX client only).

Prerequisite

From an ActiveX client application, FileXfer must be a FileTransfer Object created using the FileTransfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

See Also

[Send \(method\)](#) on page 106
[Receive \(method\)](#) on page 102
[Abort \(method\)](#) on page 94
[Status \(method\)](#) on page 110
[Reset \(method\)](#) on page 104

CrLf (property)

3270 display sessions only**Object**

FileTransfer

Syntax**AvivaBasic Macro**

```
rc% = Filetransfer.CrLf
FileTransfer.CrLf = rc%
```

ActiveX client application

```
rc% = Object1.CrLf
Object1.CrLf = crlf%
```

Description

Controls the carriage return and line feed codes for ASCII files.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Constants	Value	Description
ebxNoCrlf	0	Do not preserve CRLF (carriage return/line feed)
ebxKeepCrlf	1	Preserve CRLF

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

HostFile (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
name$ = FileTransfer.HostFile
FileTransfer.HostFile = name$
```

ActiveX client application

```
name$ = Object1.HostFile
Object1.HostFile = name$
```

Description

Name of the host file.

Return Value(s)

Returns *name\$* as String.

Value	Description
<i>name\$</i>	Is the Host file name, maximum length is 56 characters.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

Remarks

If HostFile is incorrectly set, ERR_BADFILENAME is generated at runtime.

LogRecLen (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc% = FileTransfer.LogRecLen
FileTransfer.LogRecLen = rc%
```

ActiveX client application

```
rc% = Object1.LogRecLen
Object1.LogRecLen = rc%
```

Description

File record length.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
<i>rc%</i>	Specifies the file record length, from 1 to 32760 bytes. The default value is 1.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

OnFileTransferDone

3270 display sessions only

Object

FileTransfer

Syntax

```
Sub OnFileTransferDone(status as integer, bytestransferred as Long)
  'Add handler code here
End Sub
```

Description

This is not a macro command. OnFileTransferDone is a callback function that notifies you of a file transfer operation. When this happens, the macro subsystem searches for OnFileTransferDone.

However, FileTransfer.Send(false) or FileTransfer.Receive(false) must be issued prior to OnFileTransferDone and your macro must be running at the time of the OnFileTransferDone call.

Parameters	Description
<i>status%</i>	The status of the file transfer. Refer to FileTransfer.Status method for possible values.
<i>Bytestransferred&</i>	Number of bytes transferred.
status%	Description
STATUS_COMPLETE	The call to the host has finished
STATUS_ABORTED	The call to the host has stopped
Parameters	Description

Remarks

This subroutine cannot be debugged and all breakpoints will be ignored during program execution.

Prerequisite

FileTransfer.**Send** (false) or FileTransfer.**Receive** (false)

PacketSize (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc& = FileTransfer.PacketSize
FileTransfer.PacketSize = rc&
```

ActiveX client application

```
rc& = Object1.PacketSize
Object1.PacketSize = rc&
```

Description

Retrieve or set the size (in bytes) of a packet.

Return Value(s)

Returns *rc&* as Long.

Value	Description
<i>rc&</i>	Size of the packet to be sent to the host, from 256 to 32752 bytes. The default size is 8192.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

PCFile (property)**3270 display sessions only****Object**

FileTransfer

Syntax**AvivaBasic Macro**

```
name$ = FileTransfer.PCFile
FileTransfer.PCFile = name$
```

ActiveX client application

```
name$ = Object1.PCFile
Object1.PCFile = name$
```

Description

Set or retrieve the name of your computer file.

Return Value(s)

Returns *name\$* as String.

Value	Description
<i>name\$</i>	The name of your computer file including the path, up to 255 characters long.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

Remarks

If PCFile is incorrectly set an error, ERR_BADFILENAME is generated at runtime.

Receive (method)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc% = FileTransfer.Receive([wait%])
```

ActiveX client application

```
rc% = Object1.Receive([wait%])
```

Description

Receive a file from the host. The file transfer may be synchronous (dedicated), or asynchronous (call-and-return).

Parameters	Description
<i>wait%</i>	If <i>wait%</i> is FALSE (default), then the file is received asynchronously. The file transfer command immediately returns a value.
	If <i>wait%</i> is TRUE, then the file is received synchronously. The file transfer command only returns a value when the transfer is finished.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	FileTransfer request was successful.
ERR_XFER_HOSTNOTAVAILABLE	Host is not available or a file transfer is in progress.
ERR_INVALID_PARAM	A parameter is not valid.
ERR_SYSTEM_ERROR	System error.
ERR_XFER_ABORT	The transfer has stopped. This applies when the <i>wait%</i> flag is specified as TRUE.
ERR_FAIL	Session terminating (ActiveX client only).

Prerequisite

The **host** must be correctly set prior to any FileTransfer actions.

From an ActiveX client application, FileXfer must be a FileTransfer Object created using the FileTransfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

Remarks

Use **Status** to verify whether the transfer is in progress, or if an error has occurred. If **BytesTransferred** is > 0, the file transfer has been started.

For **macro users**, the callback routine **OnFileTransferDone** notifies you when the file transfer is complete, only if the following conditions are satisfied:

- The macro script defines a subroutine named:

```
Sub OnFileTransferDone(status as Integer, bytestransferred as Long
`Code follows here
```

- The macro script has previously called FileTransfer.Send(false).
- The macro script is still running when the file transfer is completed.

WHLLAPI Reference

WHLLAPI ReceiveFile (function 91)

See Also

[Send \(method\)](#) on page 106
[Abort \(method\)](#) on page 94
[BytesTransferred \(method\)](#) on page 96
[Status \(method\)](#) on page 110
[Reset \(method\)](#) on page 104

RecFormat (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc% = FileTransfer.RecFormat
FileTransfer.RecFormat = rc%
```

ActiveX client application

```
rc% = Object1.RecFormat
Object1.RecFormat = rc%
```

Description

Sets the record format of the host file being transferred.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Constants	Value	Description
ebxUnspecify	0	Unspecified record format (default)
ebxRecFormatFixed	1	Fixed record format.
ebxRecFormatVariable	2	Variable record format.
ebxRecFormatUndefined	3	Undefined record format, (for TSO and PROGINET only.)

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

Reset (method)

3270 display sessions only**Object**

FileTransfer

Syntax**AvivaBasic Macro**

```
FileTransfer.Reset
```

ActiveX client application

```
object1.Reset
```

Description

This method resets all the properties of the FileTransfer object to default values, including the PCFile and HostFile properties.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

See Also

[Send \(method\)](#) on page 106
[Receive \(method\)](#) on page 102
[Abort \(method\)](#) on page 94
[BytesTransferred \(method\)](#) on page 96
[Status \(method\)](#) on page 110

Scheme (property)

3270 display sessions only

A file transfer scheme is a group of settings that tells Aviva how to handle the transfer of a file. Each file in the Portfolio has a transfer scheme associated with it. You can create and save new file transfer schemes, or you can customize existing ones. Schemes are stored in a scheme repository file, and can be reused by multiple sessions, and even by multiple users.

To know more about Aviva file transfer schemes, consult the Aviva online help.

Object

FileTransfer

Syntax**AvivaBasic Macro**

```
FileTransfer.Scheme = MyScheme$  
MyScheme$ = FileTransfer.Scheme
```

ActiveX client application

```
Object1.Scheme = MyScheme$  
MyScheme$ = Object1.Scheme
```

Remarks

- When you use a File Transfer scheme, the following FileTransfer properties are not used: Append, BlockSize, CrLf, LogRecLen, RecFormat, SpaceAlloc, SpaceQuantity, SpaceIncrement, Translate, and UserOptions.
- By default this property is an empty string ("").
- A valid scheme contains all the properties that are relevant to a specific type of file transfer. For example, you cannot use a Binary_TSO scheme to transfer a file to a VM host.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set FileXfer = MySession.FileTransfer
```

Send (method)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc% = FileTransfer.Send([wait%])
```

ActiveX client application

```
rc% = Object1.Send([wait%])
```

Description

Sends a file to the host in asynchronous or synchronous mode.

Parameters

If the *Wait%* flag is FALSE, use the Status property to verify if the transfer is in progress, or an error has occurred.

Parameter	Description
<i>wait%</i>	If <i>wait%</i> is FALSE (default), then the file is sent asynchronously. The file transfer command immediately returns a value.
	If <i>wait%</i> is TRUE, then the file is sent synchronously. The file transfer command only returns a value when the transfer is finished.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	File transfer request was successful.
ERR_XFER_HOSTNOTAVAILABLE	Host is not available or a file transfer is in progress.
ERR_INVALID_PARAM	A parameter is not valid.
ERR_SYSTEM_ERROR	System error.
ERR_XFER_ABORT	The transfer has stopped. This applies when the <i>wait%</i> flag is specified as TRUE
ERR_FAIL	Session is terminating (ActiveX client only).

Prerequisite

The **host** must be correctly set prior to any FileTransfer actions.

From an ActiveX client application, FileXfer must be a FileTransfer Object created using the FileTransfer method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set FileXfer = MySession.FileTransfer
```

Remarks

Use **Status** to verify whether the transfer is in progress, or if an error has occurred. If **BytesTransferred** is > 0, the file transfer has been started.

For **macro users**, the callback routine **OnFileTransferDone** notifies you when the file transfer is complete, only if the following conditions are satisfied:

- The macro script defines a subroutine named:

```
Sub OnFileTransferDone(status as Integer, bytestransferred as Long  
`Code follows here
```

- The macro script has previously called FileTransfer.Send(false).
- The macro script is still running when the file transfer is completed.

WHLLAPI Reference

WHLLAPI SendFile (function 90)

See Also

[Receive \(method\)](#) on page 102
[Abort \(method\)](#) on page 94
[BytesTransferred \(method\)](#) on page 96
[Status \(method\)](#) on page 110
[Reset \(method\)](#) on page 104

SpaceAlloc (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc% = FileTransfer.SpaceAlloc  
FileTransfer.SpaceAlloc = space%
```

ActiveX client application

```
rc% = Object1.SpaceAlloc  
Object1.SpaceAlloc = rc%
```

Description

Set the allocation type for the data set.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value. = 1.

Constants	Value	Description
	0	Unspecified allocation (default)
ebxAllocByBlocks	1	Allocation by blocks.
ebxAllocByTracks	2	Allocation by tracks.
ebxAllocByCylinders	3	Allocation by cylinders.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

SpaceIncrement (property)

3270 display sessions only**Object**

FileTransfer

Syntax**AvivaBasic Macro**

```
rc% = FileTransfer.SpaceIncrement
FileTransfer.SpaceIncrement = increment%
```

ActiveX client application

```
rc% = Object1.SpaceIncrement
Object1.SpaceIncrement = rc%
```

Description

Set the secondary space allocation. This is the amount of space allocated once the primary space allocation is completely used.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
<i>rc%</i>	Secondary space allocation. The default is 0.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

SpaceQuantity (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc% = FileTransfer.SpaceQuantity
FileTransfer.SpaceQuantity = rc%
```

ActiveX client application

```
rc% = Object1.SpaceQuantity
```

Description

Retrieve or allocate the amount of primary space. The type of primary allocation is determined by the SpaceAlloc property.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
<i>rc%</i>	The default is 0, (the host default value).

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

Status (method)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc% = FileTransfer.Status
```

ActiveX client application

```
rc% = Object1.Status
```

Description

This is a read-only method returns the status of the current file transfer.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
STATUS_NONE	No call in progress
STATUS_COMPLETE	The call to the host has finished
STATUS_INPROGRESS	There is a call in progress to the host
STATUS_ABORTED	The call to the host has stopped
STATUS_ABORT_INPROGRESS	The call to the host is being stopped
ERR_SYSTEM_ERROR	A system error was encountered.

Prerequisite

From an ActiveX client application, FileXfer must be a FileTransfer Object created using the FileTransfer method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set FileXfer = MySession.FileTransfer
```

Remarks

The file transfer works as follows:

Send or Receive with Wait: (rc% = FileTransfer.Send/Receive(true))

FileTransfer.Status returns one of the following:

- STATUS_NONE (when an error occurs before the transfer started)
- STATUS_COMPLETE (success)
- STATUS_ABORTED (an error has occurred and the transfer has stopped)

Because the transfer is synchronous, there is no need of a callback to **OnFileTransferDone**.

Send or Receive with No Wait: (`rc% = FileTransfer.Send/Receive(false)`)

`FileTransfer.Status` returns `STATUS_NONE`.

If there are no errors in the return code, then `FileTransfer.Status` returns one of the following:

- `STATUS_INPROGRESS`
- `STATUS_ABORT_INPROGRESS`
- `STATUS_COMPLETE` (success)
- `STATUS_ABORTED` (stopped by a user action or due to an error during transfer)

If callback to **OnFileTransferDone** is defined in the macro script, then the status parameter of this routine contains one of the following:

- `STATUS_COMPLETE` (success)
- `STATUS_ABORTED` (stopped by user action or due to an error in transfer)

See Also

[Send \(method\)](#) on page 106
[Receive \(method\)](#) on page 102
[Abort \(method\)](#) on page 94
[BytesTransferred \(method\)](#) on page 96
[Reset \(method\)](#) on page 104
[OnFileTransferDone](#) on page 99

TimeOut (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
rc& = FileTransfer.TimeOut  
FileTransfer.TimeOut = rc&
```

ActiveX client application

```
rc& = Object1.TimeOut  
Object1.TimeOut = rc&
```

Description

Sets the length of time (in seconds) that a program waits for a response from the host before sending an error message.

Return Value(s)

Returns *rc&* as Long.

Value	Description
<i>rc&</i>	Amount of time the program waits for a response from the host before an error message is sent, from 20 to 32767seconds. The default is 20.

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set FileXfer = MySession.FileTransfer
```

Translate (property)**3270 display sessions only****Object**

FileTransfer

Syntax**AvivaBasic Macro**

```
trans% = FileTransfer.Translate
FileTransfer.Translate = trans%
```

ActiveX client application

```
rc% = Object1.Translate
Object1.Translate = rc%
```

Description

Sets ASCII to EBCDIC translation on or off.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Constants	Value	Description
ebxNotTranslate	0	Do not translate
ebxTranslate	1	Translate
ebxDefault	2	Use the host default

Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set FileXfer = MySession.FileTransfer
```

UserOptions (property)

3270 display sessions only

Object

FileTransfer

Syntax

AvivaBasic Macro

```
FileTransfer.UserOptions = MyString$  
MyOptions$=FileTransfer.UserOptions
```

ActiveX client application

```
Object1.UserOptions = MyOptions$  
MyOptions$=Object1.UserOptions
```

Description

Lets you specify additional parameters for the file transfer. The maximum length of the string is 80 characters.

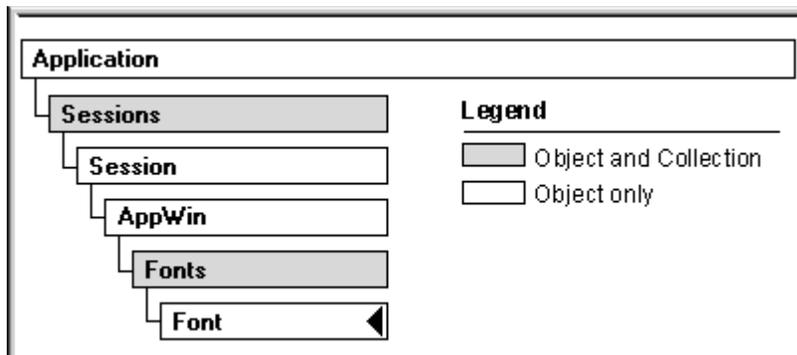
Prerequisite

From an ActiveX client application, create a File Transfer object by using the File Transfer method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set FileXfer = MySession.FileTransfer
```

Font Object

Display sessions only



This object provides processing for font manipulation on display sessions. Macro scripts and client applications can obtain the Font object reference from the AppWin object - Fonts method, and the methods of the Fonts collection object.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The predefined object name for macro scripts is "FontObj". The predefined data type for macro scripts is "Font".

Note: Due to a Microsoft limitation in Visual Basic, you cannot use Dual Interface syntax to declare the Aviva Font object. To declare the Font object, you must use the following IDispatch syntax: `Dim myFont As Object`

Font Manipulation Methods

Apply	Sets a font object to be the current host font.
--------------	---

Font Manipulation Properties

Name	Sets or retrieves the font name.
Size	Sets or retrieves the font size.
Width	Sets or retrieves the font width

Remarks

In AvivaBasic, you can manipulate this object directly. For example:

```
FontObj.Name = "Courier"
```

Or, use indirect methods by declaring your object as Font. For example:

```
Dim MyFontObject as Font
Set MyFontObject = Fonts (1)
```

For ActiveX Automation, only the following applies:

```
Dim MyFontObject as Object
Set DispSessObj = GetObject("MySession.a3d")
Set MyFontObject = DispSessObj.AppWin.Fonts (1)
```

See Also

[Fonts Object](#) on page 118
[Fonts \(method\)](#) on page 56

Apply (method)

Display sessions only

Object

Font

Syntax

AvivaBasic Macro

```
rc% = FontObj.Apply
The predefined object name for macro scripts is "FontObj".
```

ActiveX client application

```
rc% = Object1.Apply
```

Description

Use this method to apply the current font object setting. This method is used after setting one or more font properties:

Name	Sets or retrieves the font name.
Size	Sets or retrieves the font size.
Width	Sets or retrieves the font width

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFontObj = MySession.AppWin.Fonts(1)
```

Name (property)

Display sessions only

Object

Font

Syntax

AvivaBasic Macro

```
name$ = FontObj.Name
```

```
FontObj.Name=name$
```

The predefined object name for macro scripts is "FontObj".

ActiveX client application

```
name$ = Object1.Name
```

```
MyFontObj.Name=name$
```

Description

Return or set the current font name.

Return Value(s)

Returns *name\$* as a string.

Prerequisite

From an ActiveX client application, MyFontObj must be a Font object created using the Fonts method. For example:

```
Set MySession = GetObject("MySession.a3d")
```

```
Set MyFontObj = MySession.AppWin.Fonts(1)
```

Size (property)

Display sessions only

Object

Font

Syntax

AvivaBasic Macro

```
Size% = FontObj.Size
```

```
FontObj.Size=Size%
```

The predefined object name for macro scripts is "FontObj".

ActiveX client application

```
Size% = Object1.Size
```

Description

Return or set the current font size.

Return Value(s)

Returns *Size%* as an integer. This return code can be a constant or an error message value.

Prerequisite

From an ActiveX client application, MyFontObj must be a Font object created using the Fonts method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFontObj = MySession.AppWin.Fonts(1)
```

Width (property)

Display sessions only**Object**

Font

Syntax**AvivaBasic Macro**

```
MyWidth% = FontObj.Width
```

```
FontObj.Width = MyWidth%
```

The predefined object name for macro scripts is "FontObj".

ActiveX client application

```
MyWidth% = Object1.Width
```

Description

Return or set the current font width.

Return Value(s)

Returns *MyWidth%* as an integer. This return code can be a constant or an error message value.

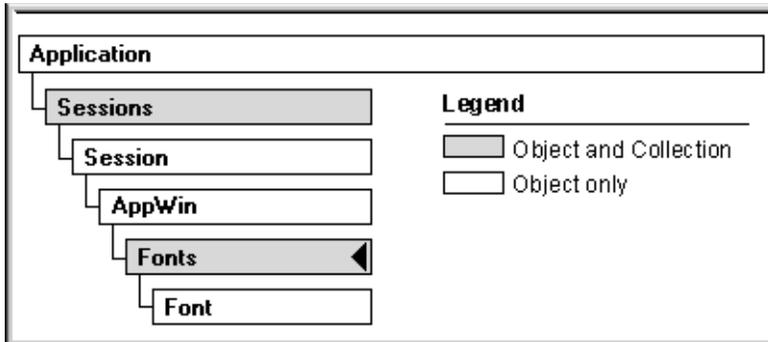
Prerequisite

From an ActiveX client application, MyFontObj must be a Font object created using the Fonts method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyFontObj = MySession.AppWin.Fonts(1)
```

Fonts Object

Display sessions only



The Fonts object allows an application to return the number of available fonts for the current session or return the object reference for a specified font.

Macro scripts and client applications can obtain the Fonts object reference from the Session object - Appwin property, or the Appwin object - Fonts method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for macro scripts is "Fonts".

Fonts Collection Methods

Current	Retrieves the currently selected font.
Item	Selects the font from the list of available fonts using its index number or name.

Fonts Collection Properties

Count	Returns the number of fonts available for the current session.
--------------	--

Remarks

In the AvivaBasic macro editor, you access the Fonts object by using any of the following commands:

```
Session.Appwin.Fonts
Appwin.Fonts
Fonts
```

For ActiveX automation, the following applies:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set MyFontObj = AppWinObj.Fonts
```

See Also

[Fonts \(method\)](#) on page 56

Count (property)

Display sessions only

Object

Fonts

Syntax

AvivaBasic Macro

rc% = **Fonts**.Count

ActiveX client application

rc% = *Object1*.Count

Description

Returns the number of available fonts.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
<i>rc%</i>	The number of available fonts.
ERR_FAIL	The session is terminating.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set MyFontObj = AppWinObj.Fonts
```

AvivaBasic Macro

Dim MyFontObj As **Font**

Current (method)

Display sessions only

Object

Fonts

Syntax

AvivaBasic Macro

```
Set MyFontObj = Fonts.Current
```

ActiveX client application

```
Set MyFontObj = Object1.Current
```

Description

Return the current font in a Font object.

Return Value(s)

Returns *Object2* as a Font Object.

Value	Meaning
FontObj	As Font (for macro)
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set MyFontObj = AppWinObj.Fonts
```

AvivaBasic Macro

```
Dim MyFontObj As Font
```

Item (method)

Display sessions only

Object

Fonts

Syntax

AvivaBasic Macro

```
Set Object2 = Fonts.Item([Index%] OR [FontName$])
```

ActiveX client application

```
Set Object2 = Object1.Item([Index%] OR [FontName$])
```

Description

From the Fonts object collection, return the specified Font as an object.

Parameters

Element	Description
<i>Index%</i>	The index number of the font.
<i>FontName\$</i>	The font name.

Return Value(s)

Returns *Object2* as a Fonts Object.

Value	Meaning
FontObj	As Font (for macro)
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

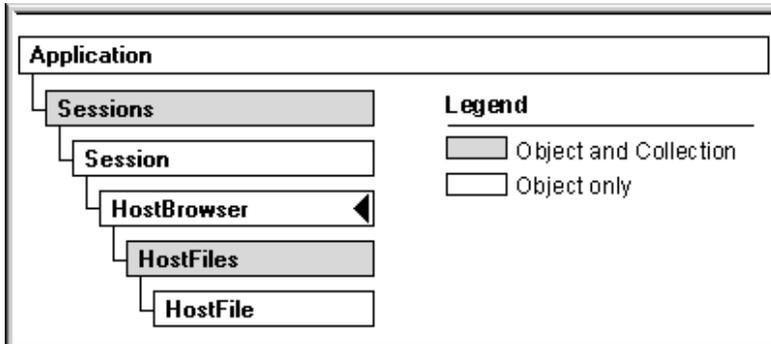
```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set MyFontObj = AppWinObj.Fonts
```

AvivaBasic Macro

```
Dim MyFontObj As Font
```

HostBrowser Object

3270 display sessions only



Host browsing lets you retrieve a list of files from the host that match a specified mask. For more information about browsing files on the host, see **About File Transfer** in the Aviva Help Topics.

Aviva Automation lets you browse a host by using the HostBrowser object. This object starts the browse and returns a HostFiles collection object. The MaxCount property specifies how many HostFile objects are contained in the HostFiles collection object (default is 10).

For example:

```
Dim HostBrowserObj as Object
Dim MySession as Object
Dim MyHostFiles1, MyHostFiles2, MyHostFiles3 as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser

' First set of 10 Hostfile objects
Set MyHostFiles1 = HostBrowserObj.HostFiles

' Next set of 10 Hostfile objects
Set MyHostFiles2 = HostBrowserObj.HostFiles

' Next set of 10 Hostfile objects
Set MyHostFiles3 = HostBrowserObj.HostFiles
```

The HostFiles collection object methods and properties let you:

- Retrieve the number of files in the list
- Retrieve a HostFile object

The HostFile object provides read-only properties that let you:

- Retrieve the name of a file
- Retrieve a file access attribute
- Retrieve a file time stamp

Macro scripts and client applications can obtain the HostBrowser object reference from the Session object - HostBrowser property.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The methods and properties of this object may only be executed from an ActiveX client application

Methods

HostFiles	Returns a HostFiles collection object of HostFile objects.
Cancel	Stops browsing (started by the HostFiles method).
Status	Retrieves the status of the HostFiles browse.
Reset	Resets all HostBrowser properties to default values.

Properties

Mask	Sets or retrieves a mask for the HostFiles browse.
MaxCount	Sets or retrieves the maximum number of HostFile objects contained in the HostFiles collection object.
TimeOut	Sets or retrieves a time-out value for HostFiles browsing

Cancel (method)

3270 display sessions only

Object

HostBrowser

Syntax

This method may only be executed from an ActiveX client application.

HostBrowserObj.Cancel

Description

Stop browsing (started by the HostFiles method).

Prerequisite

The host environment must be in file transfer mode.

From an ActiveX client application, HostBrowserObj must be a HostBrowser object created using the Session property. For example:

```
Dim HostBrowserObj as Object
Dim MySession as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
```

HostFiles (method)

3270 display sessions only

Object

HostBrowser

Syntax

This method may only be executed from an ActiveX controller application.

```
Set MyHostFiles = HostBrowserObj.Hostfiles([wait%])
```

Description

Returns an Automation collection of HostFile objects.

Parameter	Description
wait%	If <i>wait%</i> is FALSE, then the call to the host is sent asynchronously. The command returns control immediately to client application. Use the Status method to check on the status of host browsing.
	If <i>wait%</i> is TRUE (default), then the call to the host is sent synchronously. The client application must wait until the command completes before continuing.
Return Value(s)	
A HostFiles object reference	Successful.
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.
LastError	Description
ERR_INVALID_CALL	There is an invalid call.
ERR_SESSION_INHIBITED	Target field is protected or inhibited, or illegal data sent to the target field.
ERR_INVALID_PARAM	A parameter is not valid.
ERR_SYSTEM_ERROR	System error.
ERR_HOST_BROWSE_UNAVAIL	This host does not support the listing of files

Prerequisite

The host environment must be in file transfer mode.

The limit to the number of files in the collection is set by the **MaxCount** property.

The **Mask** property is used to customize the browse (filter).

The **TimeOut** property is used for synchronous calls, and sets a maximum time that a client application waits for the browsing to complete.

From an ActiveX client application, HostBrowserObj must be a HostBrowser object created using the Session property. For example:

```
Dim HostBrowserObj as Object
Dim MySession as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
```

Remarks

Use the **Status** method to find out if there are more files to browse, or if a time-out has occurred.

If Status returns ERR_MOREDATA, then repeat HostFiles to obtain the next sequence of files.

Mask (property)

3270 display sessions only

Object

HostBrowser

Syntax

This method may only be executed from an ActiveX client application.

```
FileMask$ = HostBrowserObj.Mask
HostBrowserObj.Mask = FileMask$
```

Description

Set or retrieve a mask for the HostFiles browse.

The default is NULL. Only files matching this mask value are returned. For example, to match these files using an asterisk wildcard:

```
'Match all SYS1.A* files
HostBrowserObj.Mask = "SYS1.A*"
```

```
'Match all SYS1*.LIB files
HostBrowserObj.Mask = "SYS1*.LIB"
```

For more information refer to **Host File Masks** in the Aviva Help Topics.

Prerequisite

The host environment must be in file transfer mode.

From an ActiveX client application, HostBrowserObj must be a HostBrowser object created using the Session property. For example:

```
Dim HostBrowserObj as Object
Dim MySession as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
```

MaxCount (property)

3270 display sessions only**Object**

HostBrowser

Syntax

This method may only be executed from an ActiveX client application.

```
HostBrowserObj.MaxCount = maxFiles%
maxFiles% = HostBrowserObj.MaxCount
```

Description

Set or retrieve the maximum number of HostFile objects contained in the HostFiles collection object.

Return Value(s)

Returns maxFiles% as an integer. The default value is 10

Prerequisite

The host environment must be in file transfer mode.

From an ActiveX client application, HostBrowserObj must be a HostBrowser object created using the Session property. For example:

```
Dim HostBrowserObj as Object
Dim MySession as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
```

Reset (method)

3270 display sessions only

Object

HostBrowser

Syntax

This method may only be executed from an ActiveX client application.

```
HostBrowserObj.Reset
```

Description

Reset all HostBrowser properties to default values.

Prerequisite

The host environment must be in file transfer mode.

From an ActiveX client application, HostBrowserObj must be a HostBrowser object created using the Session property. For example:

```
Dim HostBrowserObj as Object  
Dim MySession as Object  
  
Set MySession = GetObject("MySession.a3d")  
Set HostBrowserObj = MySession.HostBrowser
```

Status (method)

3270 display sessions only

Object

HostBrowser

Syntax

This method may only be executed from an ActiveX client application.

```
rc% = HostBrowserObj.Status
```

Description

Retrieve the status of the HostFiles browse.

Return Value(s)

Returns *rc%* as integer.

Value	Description
0	Successful

STATUS_NONE	No call in progress
ERR_TIMEOUT	The command cannot wait because the time-out value has been exceeded
ERR_MOREDATA	There is more data to be retrieved
STATUS_INPROGRESS	There is a call in progress to the host
STATUS_ABORT_INPROGRESS	The call to the host is being stopped
STATUS_ABORTED	The call to the host has stopped
STATUS_COMPLETE	The call to the host has stopped

Prerequisite

The host environment must be in file transfer mode. From an ActiveX client application, HostBrowserObj must be a HostBrowser object created using the Session property. For example:

```
Dim HostBrowserObj as Object
Dim MySession as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
```

TimeOut (property)

3270 display sessions only**Object**

HostBrowser

Syntax

This method may only be executed from an ActiveX client application.

```
HostBrowserObj.TimeOut = TimeOutBrowse%
TimeOutBrowse% = HostBrowserObj.TimeOut
```

Description

Set or retrieve a time-out value for HostFiles browsing.

Return Value(s)

Returns *TimeOutBrowse%* as an integer. The TimeOut value is from 20 to 32,767 seconds, and the default is 20 seconds.

Prerequisite

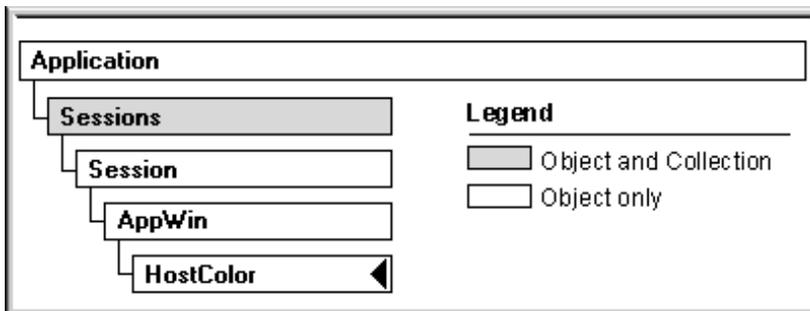
The host environment must be in file transfer mode.

From an ActiveX client application, HostBrowserObj must be a HostBrowser object created using the Session property. For example:

```
Dim HostBrowserObj as Object  
Dim MySession as Object  
  
Set MySession = GetObject("MySession.a3d")  
Set HostBrowserObj = MySession.HostBrowser
```

HostColor Object

3270 and 5250 display sessions only



Aviva supports up to 16 host colors in 3270 (see **About host colors** in the Aviva Help Topics), and up to 8 host colors in 5250. You can set how host display colors are represented on your emulation screen. The HostColor object lets you customize colors according to your needs or preferences. For example, text that displays in green on the emulation screen, can be blue or any other color you choose.

Macro scripts and client applications can obtain the HostColor object reference from the AppWin object - HostColor property.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for macro scripts is "HostColor".

Hostcolor Method

Apply	Maps all the host colors to the specified colors.
--------------	---

Hostcolor Properties

3270	5250	Color	Description
X	X	Black	Maps black to the specified color.
X	X	Blue	Maps blue to the specified color.
X		Brown	Maps brown to the specified color.
X		DarkGray	Maps dark gray to the specified color.
X		DeepBlue	Maps deep blue to the specified color.
X		Gray	Maps gray to the specified color.
X	X	Green	Maps green to the specified color.
X		Orange	Maps orange to the specified color.
X		PaleGreen	Maps pale green to the specified color.
X		PaleTurquoise	Maps pale turquoise to the specified color.
X	X	Pink	Maps pink to the specified color.
X		Purple	Maps purple to the specified color.
X	X	Red	Maps red to the specified color.
X	X	Turquoise	Maps turquoise to the specified color.
X	X	White	Maps white to the specified color.
X	X	Yellow	Maps yellow to the specified color.

About setting host colors

Host color values can be read or set by using the **RGB function**, or Hostcolor properties followed by the Hostcolor.Apply method. You can use decimal or hexadecimal values to describe color values. For a complete list of host colors and values, refer to the following table:

Note For the **RGB function**, the values are in the order, Red, Green and Blue. For hexadecimal notation, the order is reversed - &HBBGGRR.

RGB function values	Hexadecimal	Color
RGB(0,0,0)	&H000000	Black
RGB(0,0,255)	&HFF0000	Blue
RGB(128,64,64)	&H404080	Brown
RGB(192,192,192)	&HC0C0C0	Gray
RGB(0,255,0)	&H00FF00	Green
RGB(255, 200, 0)	&H0080FF	Orange
RGB(255, 175, 175)	&HFF80FF	Pink
RGB(128,0,128)	&H800080	Purple
RGB(255,0,0)	&H0000FF	Red

RGB(0,255,255)	&HFFFF00	Turquoise
RGB(255,255,255)	&HFFFFFF	White
RGB(255,255,0)	&H00FFFF	Yellow
RGB(64,64,64)	&H808080	Dark Gray
RGB(160,255,160)	&HA0FFA0	Pale Green
RGB(160,255,255)	&HFFFA0	Pale Turquoise
RGB(0,0,64)	&H800000	Deep Blue

Remarks

In the AvivaBasic macro editor, you access the Hostcolor object by using any of the following commands:

```
Session.Appwin.Hostcolor
Appwin.Hostcolor
Hostcolor
```

For ActiveX automation, the following applies:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set MyHostColObj = AppWinObj.Hostcolor
```

See Also

[HostColor \(property\)](#) on page 59

Apply (method)

3270 and 5250 display sessions only**Object**

HostColor

Syntax**AvivaBasic Macro**

```
rc% = HostColor.Apply
```

ActiveX client application

```
rc% = Object1.Apply
```

Description

You use this method after setting the HostColor properties.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
ERR_FAIL	The command was not successful.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set colorObj = AppWinObj.HostColor
```

Black (property)**3270 and 5250 display sessions only****Object**

HostColor

Syntax**AvivaBasic Macro**

```
pcCol& = HostColor.Black
```

ActiveX client application

```
pcCol& = Object1.Black
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map black to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set AppWinObj = MySession.AppWin
Set colorObj = AppWinObj.HostColor
```

Blue (property)

3270 and 5250 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Blue
```

ActiveX client application

```
pcCol& = Object1.Blue
```

To set the HostColor property to a new value for example:

```
colorObj.Blue = pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map blue to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Brown (property)

3270 and 5250 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Brown
```

ActiveX client application

```
pcCol& = Object1.Brown
```

To set the HostColor property to a new value for example:

```
colorObj.Brown = pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map brown to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

DarkGray (property)

3270 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.DarkGray
```

ActiveX client application

```
pcCol& = Object1.DarkGray
```

To set the HostColor property to a new value for example:

```
colorObj.DarkGray= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map DarkGray to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

DeepBlue (property)

3270 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.DeepBlue
```

ActiveX client application

```
pcCol& = Object1.DeepBlue
```

To set the HostColor property to a new value for example:

```
colorObj.DeepBlue= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map DeepBlue to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Gray (property)

3270 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Gray
```

ActiveX client application

```
pcCol& = Object1.Gray
```

To set the HostColor property to a new value for example:

```
colorObj.Gray= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map Gray to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Green (property)

3270 and 5250 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Green
```

ActiveX client application

```
pcCol& = Object1.Green
```

To set the HostColor property to a new value for example:

```
colorObj.Green = pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map green to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Orange (property)

3270 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Orange
```

ActiveX client application

```
pcCol& = Object1.Orange
```

To set the HostColor property to a new value for example:

```
colorObj.Orange= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map orange to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

PaleGreen (property)

3270 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.PaleGreen
```

ActiveX client application

```
pcCol& = Object1.PaleGreen
```

To set the HostColor property to a new value for example:

```
colorObj.DeepBlue= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map PaleGreen to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

PaleTurquoise (property)

3270 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.PaleTurquoise
```

ActiveX client application

```
pcCol& = Object1.PaleTurquoise
```

To set the HostColor property to a new value for example:

```
colorObj.DeepBlue= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map PaleTurquoise to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Pink (property)

3270 and 5250 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Pink
```

ActiveX client application

```
pcCol& = Object1.Pink
```

To set the HostColor property to a new value for example:

```
colorObj.pink = pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map pink to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Purple (property)

3270 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Purple
```

ActiveX client application

```
pcCol& = Object1.Purple  
colorObj.Purple= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map Purple to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Red (property)

3270 and 5250 display sessions only**Object**

HostColor

Syntax**AvivaBasic Macro**

```
pcCol& = HostColor.Red
```

ActiveX client application

```
pcCol& = Object1.Red
```

To set the HostColor property to a new value for example:

```
colorObj.Red = pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map red to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor.
```

RGB (function)

3270 display sessions only

Syntax

`rc& = RGB(red%, green%, blue%)`

Description

Combines the *red%*, *green%* and *blue%* color components into a long integer that represents an RGB color value.

Parameters	Description
<i>red%</i>	Integer (0-255) representing the red component of the color.
<i>green%</i>	Integer (0-255) representing the green component of the color.
<i>blue%</i>	Integer (0-255) representing the blue component of the color.

Return Value(s)

Returns *rc&* as an RGB color value, type Long.

Parameters

Note: For this function, the values are in the order, Red, Green and Blue. For hexadecimal notation, the order is reversed - `&HBBGRR`.

RGB function values	Hexadecimal	Color
RGB(0,0,0)	&H000000	Black.
RGB(0,0,255)	&HFF0000	Blue
RGB(128,64,64)	&H404080	Brown
RGB(192,192,192)	&HC0C0C0	Gray
RGB(0,255,0)	&H00FF00	Green
RGB(255, 200, 0)	&H0080FF	Orange
RGB(255, 175, 175)	&HFF80FF	Pink
RGB(128,0,128)	&H800080	Purple
RGB(255,0,0)	&H0000FF	Red
RGB(0,255,255)	&HFFFF00	Turquoise
RGB(255,255,255)	&HFFFFFF	White
RGB(255,255,0)	&H00FFFF	Yellow
RGB(64,64,64)	&H808080	Dark Gray
RGB(160,255,160)	&HA0FFA0	Pale Green
RGB(160,255,255)	&HFFFA0	Pale Turquoise
RGB(0,0,64)	&H800000	Deep Blue

Remarks

If the value for any argument exceeds 255, it is assumed to be 255.

See Also

[Apply \(method\)](#) on page 132

Turquoise (property)

3270 and 5250 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Turquoise
```

ActiveX client application

```
pcCol& = Object1.Turquoise
```

To set the HostColor property to a new value for example:

```
colorObj.Turquoise = pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map turquoise to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

White (property)

3270 and 5250 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.White
```

ActiveX client application

```
pcCol& = Object1.White
```

To set the HostColor property to a new value for example:

```
colorObj.White= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map white to the specified color.

Return Value(s)

Returns pcCol& as a Long.

Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

Yellow (property)

3270 and 5250 display sessions only

Object

HostColor

Syntax

AvivaBasic Macro

```
pcCol& = HostColor.Yellow
```

ActiveX client application

```
pcCol& = Object1.Yellow
```

To set the HostColor property to a new value for example:

```
colorObj.Yellow= pcCol&  
rc% = colorObj.apply
```

Description

Retrieve the current color value, or map yellow to the specified color.

Return Value(s)

Returns pcCol& as a Long.

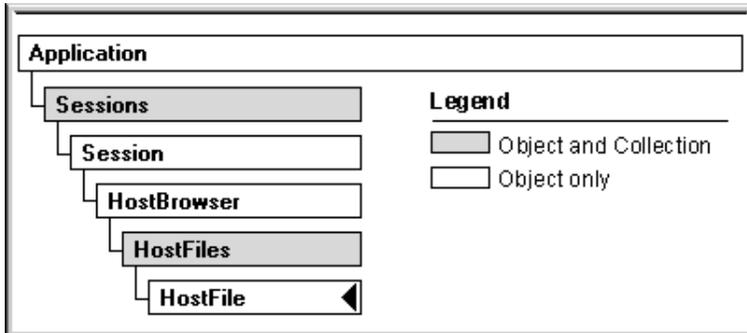
Prerequisite

From an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set AppWinObj = MySession.AppWin  
Set colorObj = AppWinObj.HostColor
```

HostFile Object

3270 display sessions only



Host browsing lets you retrieve a list of files from the host that match a specified mask. For more information about browsing files on the host, see **About File Transfer** in the Aviva Help Topics.

Aviva Automation lets you browse a host by using the HostBrowser object. This object starts the browse and returns a HostFiles collection object. The MaxCount property specifies how many HostFile objects are contained in the HostFiles collection object (default is 10).

For example:

```
Dim HostBrowserObj as Object
Dim MySession as Object
Dim MyHostFiles1, MyHostFiles2, MyHostFiles3 as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser

' First set of 10 Hostfile objects
Set MyHostFiles1 = HostBrowserObj.HostFiles

' Next set of 10 Hostfile objects
Set MyHostFiles2 = HostBrowserObj.HostFiles

' Next set of 10 Hostfile objects
Set MyHostFiles3 = HostBrowserObj.HostFiles
```

The HostFiles collection object methods and properties let you:

- Retrieve the number of files in the list.
- Retrieve a HostFile object

The HostFile object provides read-only properties that let you:

- Retrieve the name of a file.
- Retrieve a file access attribute
- Retrieve a file time stamp

Macro scripts and client applications can obtain the HostFile object reference from the HostFiles object - Item method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The properties of this object may only be executed from an ActiveX client application

Properties

Name	Retrieves a host file name.
Attribute	Retrieves the attribute of a host file.
TimeStamp	Retrieves the time-stamp of a host file.

Attribute (property)

3270 display sessions only

Object

HostFile

Syntax

This method may only be executed from an ActiveX client application.

HostFileAttrib% = *HostFileObj*.Attribute

Description

Retrieve the attribute of a host file.

Return Value(s)

Returns *HostFileAttrib%* as an Integer. This value may be processed as an OR value. For example:

the value 5 is attribute 4 and attribute 1, (container with read-only access).

Value	Meaning
1	Container
2	No access
4	Read
8	Write
16	Unknown

Prerequisite

The host environment must be in file transfer mode.

From an ActiveX client application, *HostFileObj* must be a HostFile object created using the HostFiles, Item method. For example:

```
Dim MySession as Object
Dim HostBrowserObj as Object
Dim MyHostFiles as Object
Dim MyFile as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
Set MyHostFiles = HostBrowserObj.Hostfiles
Set MyFile = MyHostFiles.Item(1)'Retrieve first file in the list
```

Name (property)

3270 display sessions only

Object

HostFile

Syntax

This method may only be executed from an ActiveX client application.

```
HostFileName$ = HostFileObj.Name
```

Description

Retrieve a host file name.

Return Value(s)

Returns *HostFileName\$* as a String.

Prerequisite

From an ActiveX client application, *HostFileObj* must be a HostFile object created using the HostFiles, Item method. For example:

```
Dim MySession as Object
Dim HostBrowserObj as Object
Dim MyHostFiles as Object
Dim MyFile as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
Set MyHostFiles = HostBrowserObj.Hostfiles
Set MyFile = MyHostFiles.Item(1)'Retrieve first file in the list
```

PrintFile (function)

3270 display sessions only

Syntax

```
rc& = PrintFile(filename$)
```

Description

This function will print an ASCII file as specified by filename\$ to the default printer. The text will be printed as 132 columns, 60 lines per page.

Parameters

Element	Description
filename\$	Name of the file to print.

Return Value(s)

Returns rc& as a Long value.

Value	Description
0	Success
ERR_CANTOPENFILE	Either the file is not found or an error has occurred when opening the file.

TimeStamp (property)

3270 display sessions only

Object

HostFile

Syntax

This method may only be executed from an ActiveX client application.

```
FileTimeStamp$ = HostFileObj.TimeStamp
```

Description

Retrieve the time-stamp of a host file.

Return Value(s)

Returns *FileTimeStamp\$* as string, or Date for Visual Basic. This string represents date and time formatted according to the current national language (locale Id).

Remarks

For users of Microsoft Foundation Classes (MFC), the COleDateTime.ParseDateTime() function converts *FileTimeStamp\$* to date and time values.

Prerequisite

The host environment must be in file transfer mode.

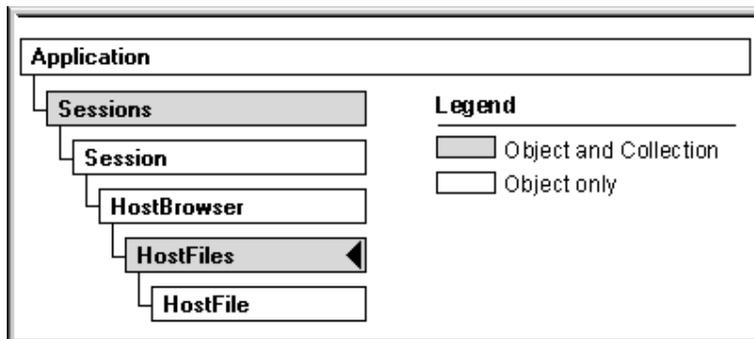
From an ActiveX client application, *HostFileObj* must be a HostFile object created using the HostFiles, Item method. For example:

```
Dim MySession as Object
Dim HostBrowserObj as Object
Dim MyHostFiles as Object
Dim MyFile as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
Set MyHostFiles = HostBrowserObj.Hostfiles
Set MyFile = MyHostFiles.Item(1)'Retrieve first file in the list
```

HostFiles Object

3270 display sessions only



Host browsing lets you retrieve a list of files from the host that match a specified mask. For more information about browsing files on the host, see **About File Transfer** in the Aviva Help Topics.

Aviva Automation lets you browse a host by using the HostBrowser object. This object starts the browse and returns a HostFiles collection object. The MaxCount property specifies how many HostFile objects are contained in the HostFiles collection object (default is 10).

For example:

```

Dim HostBrowserObj as Object
Dim MySession as Object
Dim MyHostFiles1, MyHostFiles2, MyHostFiles3 as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser

' First set of 10 Hostfile objects
Set MyHostFiles1 = HostBrowserObj.HostFiles

' Next set of 10 Hostfile objects
Set MyHostFiles2 = HostBrowserObj.HostFiles

' Next set of 10 Hostfile objects
Set MyHostFiles3 = HostBrowserObj.HostFiles
  
```

The HostFiles collection object methods and properties let you:

- Retrieve the number of files in the list.
- Retrieve a HostFile object

The HostFile object provides read-only properties that let you:

- Retrieve the name of a file.
- Retrieve a file access attribute
- Retrieve a file time stamp

Macro scripts and client applications can obtain the HostFiles object reference from the HostBrowser object - HostFiles method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The methods and properties of this object may only be executed from an ActiveX client application

Methods

Item	Returns a HostFile object reference from the Hostfiles collection object.
-------------	---

Properties

Count	Retrieves the number of files contained in the HostFiles collection object.
--------------	---

Count (property)

3270 display sessions only

Object

HostFiles

Syntax

This method may only be executed from an ActiveX client application.

HostFilesObj.Count = MyHostFiles%

Description

Retrieve the number of files contained in the HostFiles collection object.

Return Value(s)	
<i>MyHostFiles%</i>	as Integer.
LastError	Description
ERR_FAIL	A system error occurred.

Prerequisite

The host environment must be in file transfer mode.

From an ActiveX client application, HostFilesObj must be a HostFiles object created using the HostBrowser, HosFiles method. For example:

```
Dim MySession as Object
Dim HostBrowserObj as Object
Dim MyHostFiles as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
Set MyHostFiles = HostBrowserObj.Hostfiles
```

Item (method)

3270 display sessions only

Object

HostFiles

Syntax

This method may only be executed from an ActiveX client application.

```
Set MyHostFile = HostFilesObj.Item([index$]
OR
Set MyHostFile = HostFilesObj.Item[FileName$])
```

Description

Return a HostFile object reference from the HostFiles collection object.

Parameters	Description
<i>index%</i>	The index (integer) of a host file in the HostFiles collection.
<i>FileName\$</i>	A file name in the HostFiles collection.
Return Value(s)	
A HostFile object reference	Successful.
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

The host environment must be in file transfer mode.

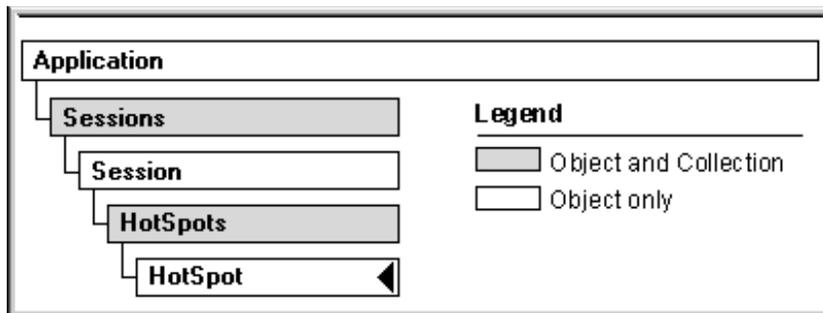
From an ActiveX client application, HostFilesObj must be a HostFiles object created using the HostBrowser, HosfFiles method. For example:

```
Dim MySession as Object
Dim HostBrowserObj as Object
Dim MyHostFiles as Object

Set MySession = GetObject("MySession.a3d")
Set HostBrowserObj = MySession.HostBrowser
Set MyHostFiles = HostBrowserObj.Hostfiles
```

HotSpot Object

Display sessions only



This object provides the properties to modify Aviva hotspots. A hotspot is a pre-defined screen element—a word or phrase generated by the host—that is activated to accept user input in the form of mouse clicks or touch screen gestures. Each Aviva session can have its own set of hotspots that you customize according to your needs and preferences. Each Aviva session can have its own set of hotspots that you customize according to your needs and preferences.

Client applications can obtain the HotSpot object reference from the Session object - HotSpots method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The properties of this object may only be executed from an ActiveX client application

Properties

Active	Sets or retrieves the state of a Hotspot.
AsButton	Sets or retrieves the button display state of a Hotspot.
MatchCase	Sets or retrieves the case sensitivity state of a Hotspot.
Name	Sets or retrieves the name of a Hotspot object .

Remarks

For ActiveX automation, the following applies:

```
Dim MyHotSpotObj as Object
Dim MySession as Object
Dim MyHotSpots as Object

Set MySession = GetObject("MySession.a3d")
Set MyHotSpots = MySession.HotSpots()
Set MyHotSpotObj = MyHotSpots.item(1)
```

Active (property)

Display sessions only

Object

HotSpot

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.Active = Bool  
Bool = Object1.Active
```

Description

Set or retrieve the state of a Hotspot.

Return Value(s)

Returns *Bool* as Boolean.

Element	Description
FALSE(Default)	Not active.
TRUE	Active.

Prerequisite

From an ActiveX client application, create a HotSpots object by using the Session HotSpots method and then create a Hotspot object by using the HotSpots Item method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyHotSpots = MySession.HotSpots()  
Set MyHotSpotObj = MyHotSpotsObj.item(1)
```

AsButton (property)

Display sessions only

Object

HotSpot

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.AsButton= Bool  
Bool = Object1.AsButton
```

Description

Set or retrieve the button display state of a Hotspot.

Return Value(s)

Returns *Bool* as Boolean.

Element	Description
FALSE(Default)	Do not display as a button.
TRUE	Display as a button.

Prerequisite

From an ActiveX client application, create a HotSpots object by using the Session HotSpots method and then create a Hotspot object by using the HotSpots Item method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyHotSpots = MySession.HotSpots()
Set MyHotSpotObj = MyHotSpotsObj.item(1)
```

MatchCase (property)

Display sessions only**Object**

HotSpot

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.MatchCase = Bool
Bool = Object1.MatchCase
```

Description

Set or retrieve the case sensitivity state of a Hotspot. You can set any Hotspot's "text to match" string as case sensitive.

Return Value(s)

Returns *Bool* as Boolean.

Element	Description
FALSE(Default)	Do not set hotspot as case sensitive.
TRUE	Set hotspot as case sensitive.

Prerequisite

From an ActiveX client application, create a HotSpots object by using the Session HotSpots method and then create a Hotspot object by using the HotSpots Item method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyHotSpots = MySession.HotSpots()
Set MyHotSpotObj = MyHotSpotsObj.item(1)
```

Name (property)

Display sessions only

Object

HotSpot

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.Name = HotSpotName$  
HotSpotName$ = Object1.Name
```

Description

Set or retrieve the name of a Hotspot object.

Return Value(s)

Returns *HotSpotName\$* as a String.

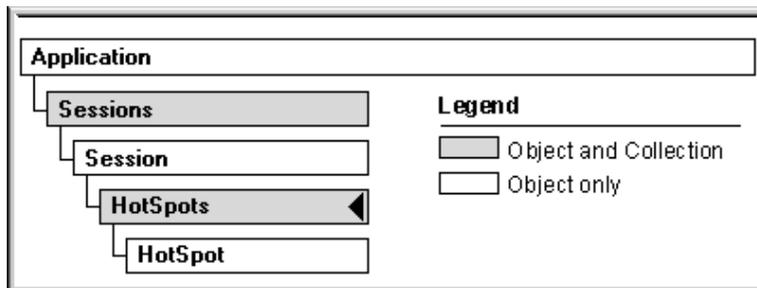
Prerequisite

From an ActiveX client application, create a HotSpots object by using the Session HotSpots method and then create a Hotspot object by using the HotSpots Item method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyHotSpots = MySession.HotSpots()  
Set MyHotSpotObj = MyHotSpotsObj.item(1)
```

HotSpots Object

Display sessions only



A hotspot is a pre-defined screen element—a word or phrase generated by the host—that is activated to accept user input in the form of mouse clicks or touch screen gestures. Each Aviva session can have its own set of hotspots that you customize according to your needs and preferences.

A HotSpots object is a collection of HotSpots. Macro scripts and client applications can obtain the HotSpots object reference from the Session object - HotSpots method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The methods and properties of this object may only be executed from an ActiveX client application.

HotSpots Collection Methods

Item	Retrieve a HotSpot object from a collection of HotSpots.
Remove	Remove a HotSpot object from a collection of HotSpots.
Load	Retrieve a HotSpots object.

HotSpots Collection Properties

Count	Retrieve the number of HotSpot objects from a collection of HotSpots.
--------------	---

Remarks

For ActiveX automation, the following applies:

```
Dim MySession as Object
Dim MyHotSpots as Object
```

```
Set MySession = GetObject("MySession.a3d")
Set MyHotSpots = MySession.HotSpots()
```

Count (property)

Display sessions only

Object

HotSpots

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = Object1.Count
```

Description

Retrieves the number of HotSpot objects from a collection of HotSpots.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Prerequisite

From an ActiveX client application, create a HotSpots object by using the HotSpots() method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyHotSpots = MySession.HotSpots()
```

Item (method)

Display sessions only

Object

HotSpots

Syntax

This method may only be executed from an ActiveX controller application.

```
Set Object2 = Object1.Item([index%])  
or  
Set Object2 = Object1.Item([hotspotname$])
```

Description

Retrieves a HotSpot object from a collection of HotSpots.

Parameters	Description
<i>index%</i>	The HotSpot index number.
<i>hotspotname\$</i>	The HotSpot name

Return Value(s)

Returns *object2* as a HotSpot object.

Value	Description
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_INVALID_PARAM	There is an invalid parameter.
ERR_OUTOFMEMORY	There is not enough memory to create this object.
ERR_FAIL	Cannot remove the object specified by index% or hotspotname\$.

Prerequisite

From an ActiveX client application, create a HotSpots object by using the HotSpots() method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyHotSpots = MySession.HotSpots()
```

Load (method)**Display sessions only****Object**

HotSpots

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = Object1.Load(collectionName$, fileName$)
```

Description

Retrieves a HotSpots object.

Parameters	Description
<i>collectionName\$</i>	HotSpots collection name.
<i>fileName\$</i>	The name of the session or repository file where the collection of hotspots reside. If you specify a repository file, you must include a <i>collectionName\$</i> . Aviva Property Manager lets you see all the HotSpots in a repository file. If you specify a session file, then set <i>collectionName\$</i> = "".

Return Value(s)

Returns *rc%* as Integer.

Value	Description
0	The command was successful.
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_SYSTEM_ERROR	The command failed due to a system error
ERR_FILENOTFOUND	The file specified by fileName\$ cannot be found
ERR_CANTOPENFILE	The file specified by fileName\$ cannot be opened
ERR_FILEACCESSERROR	There was an error accessing the file

Prerequisite

From an ActiveX client application, create a HotSpots object by using the HotSpots() method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyHotSpots = MySession.HotSpots()
```

Remarks

You can replace your current session's hotspots by loading any hotspot collection from a session file or a repository file.

Remove (method)

Display sessions only**Object**

HotSpots

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = Object1.Remove([index%])
or
rc% = Object1.Remove([hotspotname$])
```

Description

Removes a HotSpot object from a collection of HotSpots.

Parameters	Description
<i>index%</i>	The HotSpot index number.
<i>hotspotname\$</i>	The HotSpot name.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The command was successful.
Nothing	An error occurred, the LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_INVALID_PARAM	There is an invalid parameter.
ERR_OUTOFMEMORY	There is not enough memory to create this object.
ERR_FAIL	Cannot remove the object specified by index% or hotspotname\$.

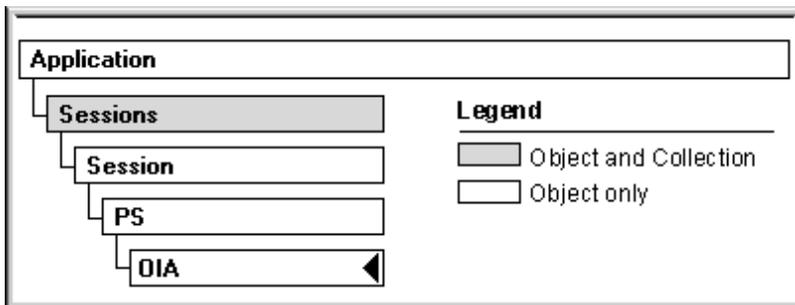
Prerequisite

From an ActiveX client application, create a HotSpots object by using the HotSpots() method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyHotSpots = MySession.HotSpots()
```

OIA Object

3270 and 5250 display sessions only



This object provides information about the OIA line. Macro scripts and Controller applications can obtain the OIA object reference from the PS object - OIA method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for Macro Scripts is "OIA".

3270	5250	Properties	Description
X		APLMode	Returns the state of the APL Mode: On or Off.
X		CommCheck	Indicates status on host communications.
X		GraphicCursorMode	Returns the state of the Graphic Cursor Mode: On or Off.
X	X	InputInhibit	Indicates whether session is Input Inhibited.
X	X	InputInhibitState	Retrieves the current value of the Input Inhibit state.
X	X	InsertMode	Returns the state of the Insert Mode: On or Off.
X		MachineCheck	Indicates whether session is working properly or not.
X	X	Ownership	Returns the ownership of the session.

X		ProgramCheck	Indicates whether a programming error is detected in the data received by the control unit.
	X	MessageWaiting	Indicates whether a message is waiting or not waiting.
	X	SystemAvailable	Indicates whether the system is available or not available.

Remarks

In the Macro editor, you access the OIA object as follows:

```
OIA
or
PS.OIA
```

For ActiveX automation, the following applies:

```
Dim MySession as Object
Dim MyPSObj as Object
Dim MyOIA as Object

Set MySession = GetObject("MySession.a3d")
Set MyPSObj = MySession.PS
Set MyOIA = MyPSObj.OIA
```

See Also

[OIA \(property\)](#) on page 192

APLMode (property)

3270 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

Bool = **OIA**.APLMode

ActiveX controller application

Bool = *object1*.APLMode

Description

Determines whether the APLMode is on or off. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	The keyboard is not in APL mode.
TRUE	The keyboard is in APL mode. APL characters and functions are available.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

CommCheck (property)

3270 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

```
Bool = OIA.CommCheck
```

ActiveX controller application

```
Bool = object1.CommCheck
```

Description

Returns the status of the host communication. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	There is no problem with the communication line.
TRUE	There is a problem with the communication line. This property is set when attempting to communicate to host and a communication error reminder is present in the operator information area.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

GraphicCursorMode (property)

3270 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

Bool = **OIA**.GraphicCursorMode

ActiveX controller application

Bool = *object1*.GraphicCursorMode

Description

Determines whether graphic cursor mode is on or off. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	The graphic cursor is off.
TRUE	The graphic cursor is on.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

InputInhibit (property)

3270 and 5250 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

Bool = **OIA**.InputInhibit

ActiveX controller application

Bool = *object1*.InputInhibit

Description

Determines whether the session's input is inhibited or not inhibited. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	The input from the keyboard,mouse or touch screen gesture is allowed.
TRUE	The input from the keyboard , mouseor touch screen gesture is not allowed.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

InputInhibitState (property)

3270 and 5250 display sessions only

Object

OIA

Syntax

ActiveX controller application

```
InhibitState% = object1.InputInhibitState
```

Description

This is a read-only property indicates the current value of Input Inhibit.

Return Value(s)

Returns *InhibitState%* as an Integer.

Value	Mnemonic	Description
0	II_CANTYPE	The session is not connected.
1	II_WAIT	Host needs more time to respond
2	II_LOCALERR 5250 only	Cannot enter data. Keyboard is locked
2	II_SYSTEM 3270 only	Cannot enter data. Keyboard is locked
3	II_PROG	Error in the data from the host.
10	II_MACHINE	Problem with display station or control unit

11	II_OFF	
The following need Reset to unlock keyboard:		
4	II_WHAT	Display station did not accept last input.
5	II_MINUSF	Unavailable keyboard function was requested
6	II_MINUSFNOOP	Invalid operation when cursor is not in an input field or a detectable field.
7	II_GOELSEWHERE	
8	II_MORETHAN	Input field has no space for characters
9	II_NUM	Non numeric character in a numeric field.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

InsertMode (property)

3270 and 5250 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

```
Bool = OIA.InsertMode
```

ActiveX controller application

```
Bool = object1.InsertMode
```

Description

Determines whether insert mode is on or off. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	Insert mode is turned off.
TRUE	Insert mode is turned on.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

MachineCheck (property)

3270 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

```
Bool = OIA.MachineCheck
```

ActiveX controller application

```
Bool = object1.MachineCheck
```

Description

Determines whether the session is working correctly. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	There is no machine check.
TRUE	There is a machine check, and the emulated terminal is not working properly.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

MessageWaiting (property)

5250 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

Bool = OIA.MessageWaiting

ActiveX controller application

Bool = *object1*.MessageWaiting

Description

Indicates whether a message is waiting or not waiting. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	There is a message waiting.
TRUE	There is no message waiting.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

Ownership (property)

3270 and 5250 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

owner% = OIA.Ownership

ActiveX controller application

owner% = *object1*.Ownership

Description

This read-only property indicates who has ownership of the current session. The *owner%* value can be a constant or an error message value.

Return Value(s)

Returns *owner%* as an integer.

Constants	Value	Description
ebxLU_NoSession	1	The session is not connected.
ebxLU_UnOwned	2	The session is active and there is no program running.
ebxLU_SysOp	3	The session is owned by the System Services Control Point.
ebxLU_MyJob	4	The session is owned by a user's program.
Error Messages: Click message to see value		
ERR_FAIL		Session terminating (ActiveX Automation only).
Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.		

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

ProgramCheck (property)

3270 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

```
Bool = OIA.ProgramCheck
```

ActiveX controller application

```
Bool = object1.ProgramCheck
```

Description

Determines whether a programming error is detected in the data received by the control unit. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	There is no program check.
TRUE	There is a program check, and there is a programming error in the data from the host.

Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

SystemAvailable (property)

5250 display sessions only

Object

OIA

Syntax

AvivaBasic Macro

```
Bool = OIA.SystemAvailable
```

ActiveX controller application

```
Bool = object1.SystemAvailable
```

Description

Indicates whether the system is available or not available. This is a read-only property.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	System is not available.
TRUE	System is available.

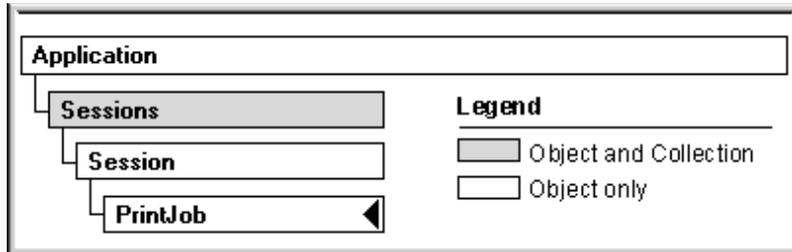
Prerequisites

When executing from an ActiveX controller application, create an OIA object by using the PS property. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyOIAObj = MySession.PS.OIA
```

PrintJob Object

Printer sessions only



This object allows actions to be performed on a print job. Macro scripts and client applications can obtain the PrintJob object reference from the Session object - PrintJob property.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The methods of this object may only be executed from an ActiveX client application.

Methods

The application object encompasses the following methods:

Cancel	Cancels a session print job.
Pause	Pauses a session print job.
Resume	Resumes a session print job that was paused.
State	Retrieves the current status of a session print job.
PA1	Performs host function key PA1 (3270 only).
PA2	Performs host function key PA2 (3270 only).

See Also

[PrintJob \(property\)](#) on page 249

Cancel (method)

Printer sessions only

Object

PrintJob

Syntax

This method may only be executed from an ActiveX client application.

rc% = *Object1*.Cancel

Description

This method cancels the session print job.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
ERR_FAIL	Command failed or session is terminating.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
0	No errors were encountered.

Prerequisite

Before executing the Cancel method, your script or program must have **SetSharing()** set to SHARE_WITH_ALL on the Session object from which PrintJob object was created. It is required that SHARE_WITH_ALL be set at least once per session.

From an ActiveX client application, create a Printer Session object by using the GetObject command. For example:

```
Set MySession = GetObject("MyPrinter.a3p")
```

See Also

[PA1 \(method\)](#) on page 175
[PA2 \(method\)](#) on page 176
[Pause \(method\)](#) on page 177
[Resume \(method\)](#) on page 178
[State \(method\)](#) on page 179

PA1 (method)

3270 printer sessions only**Object**

PrintJob

Syntax

This method may only be executed from an ActiveX client application.

```
rc% = Object1.PA1
```

Description

This method sends a PA1 key to the current print job.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
ERR_FAIL	Command failed or session is terminating.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
0	No errors were encountered.

Prerequisite

Before executing the Cancel method, your script or program must have **SetSharing()** set to SHARE_WITH_ALL on the Session object from which PrintJob object was created. It is required that SHARE_WITH_ALL be set at least once per session.

From an ActiveX client application, create a Printer Session object by using the GetObject command. For example:

```
Set MySession = GetObject("MyPrinter.a3p")
```

See Also

- [PA2 \(method\)](#) on page 176
- [Pause \(method\)](#) on page 177
- [Resume \(method\)](#) on page 178
- [Cancel \(method\)](#) on page 174
- [State \(method\)](#) on page 179

PA2 (method)

3270 printer sessions only

Object

PrintJob

Syntax

This method may only be executed from an ActiveX client application.

```
rc% = Object1.PA2
```

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
ERR_FAIL	Command failed or session is terminating.
ERR_SYSTEM_ERROR	A system error was encountered.

ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
0	No errors were encountered.

Prerequisite

Before executing the Cancel method, your script or program must have **SetSharing()** set to SHARE_WITH_ALL on the Session object from which PrintJob object was created. It is required that SHARE_WITH_ALL be set at least once per session.

From an ActiveX client application, create a Printer Session object by using the GetObject command. For example:

```
Set MySession = GetObject("MyPrinter.a3p")
```

See Also

- [PA1 \(method\)](#) on page 175
- [Pause \(method\)](#) on page 177
- [Resume \(method\)](#) on page 178
- [Cancel \(method\)](#) on page 174
- [State \(method\)](#) on page 179

Pause (method)

Printer sessions only

Object

PrintJob

Syntax

This method may only be executed from an ActiveX client application.

```
rc% = Object1.Pause
```

Description

This method pauses the current session print job.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
ERR_FAIL	Command failed or session is terminating.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
0	No errors were encountered.

Prerequisite

Before executing the Cancel method, your script or program must have **SetSharing()** set to SHARE_WITH_ALL on the Session object from which PrintJob object was created. It is required that SHARE_WITH_ALL be set at least once per session.

From an ActiveX client application, create a Printer Session object by using the GetObject command. For example:

```
Set MySession = GetObject("MyPrinter.a3p")
```

See Also

[PA1 \(method\)](#) on page 175
[PA2 \(method\)](#) on page 176
[Resume \(method\)](#) on page 178
[Cancel \(method\)](#) on page 174
[State \(method\)](#) on page 179

Resume (method)

Printer sessions only**Object**

PrintJob

Syntax

This method may only be executed from an ActiveX client application.

```
rc% = Object1.Resume
```

Description

This method resumes the current session print job, (see **Pause**)

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
ERR_FAIL	Command failed or session is terminating.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
0	No errors were encountered.

Prerequisite

Before executing the Cancel method, your script or program must have **SetSharing()** set to SHARE_WITH_ALL on the Session object from which PrintJob object was created. It is required that SHARE_WITH_ALL be set at least once per session.

From an ActiveX client application, create a Printer Session object by using the GetObject command. For example:

```
Set MySession = GetObject("MyPrinter.a3p")
```

See Also

- [PA1 \(method\)](#) on page 175
- [PA2 \(method\)](#) on page 176
- [Pause \(method\)](#) on page 177
- [Cancel \(method\)](#) on page 174
- [State \(method\)](#) on page 179

State (method)

Printer sessions only

Object

PrintJob

Syntax

This method may only be executed from an ActiveX client application.

```
rc% = Object1.State
```

Description

This method returns the state of the current print job.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Constants	Value	Description
ebxPrinterIdle	0	Idle
ebxPrinterPrinting	1	Printing
ebxPrinterPaused	2	Paused
ERR_FAIL		Command failed or session is terminating.

Prerequisite

From an ActiveX client application, create a Printer Session object by using the GetObject command. For example:

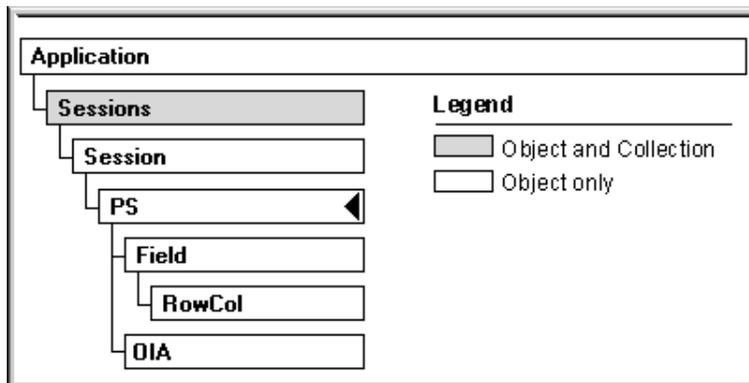
```
Set MySession = GetObject("MyPrinter.a3p")
```

See Also

- [PA1 \(method\)](#) on page 175
- [PA2 \(method\)](#) on page 176
- [Pause \(method\)](#) on page 177
- [Resume \(method\)](#) on page 178
- [Cancel \(method\)](#) on page 174

PS Object

Display sessions only



This object provides processing for cursor position and host data manipulation. Macro scripts and Controller applications can obtain the PS object reference from the Session object - PS property.

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation			VT emulation		
Model	Rows	Columns	Model	Rows	Columns	Mode	Rows	Columns
2	24	80	Normal	24	80	80 cols	24	80
3	32	80	Alternate	27	132	132 cols	27	132
4	43	80						
5	27	132						
11	62	160						

5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for Macro Scripts is "PS".

Cursor Position Manipulation Methods

GetCursorLocation	Returns the cursor location in row and column values.
MaxRowColumn	Retrieves the maximum row and column position in the presentation space.
RowColToPosition	Converts row and column into a host presentation space position value.
SetCursorLocation (3270 and 5250 only)	Sets the cursor position with specified row and column values.
WaitCursorAt	Waits for the cursor to move to a specified row and column position.

Screen Data Manipulation Methods

FindString	Finds the specified string.
RetrieveKey	Gets the intercepted key stroke entered by a user.
GetData	Copies the entire or a portion of the presentation space into a string buffer.
QueryHostUpdate	Checks if any host update occurred in the presentation space. OIA or both presentation space and OIA are checked.
QueryHostUpdateWithWait (3270 and 5250 only)	Checks if any host update occurred in the presentation space, the OIA, or in both. Returns from the routine when the update occurs or when a time-out value expires.
Reset	Reinitializes all properties of the PS object to default values.
SendString	Sends a string or key to a specified position.
SetData	Copies the user-defined string to a specified location.
StartKeyIntercept	Starts to intercept a key entered by a user.
StartHostNotification	Starts notification on PS, OIA, or both PS and OIA.
StopKeyIntercept	Stops intercepting key entered by a user.
StopHostNotification	Resets the StartHostNotification request.
WaitForString	Waits for the specified string to appear before executing the next statement.
WaitHostSettle	Waits for the host to be ready for input.

Object Creation Methods

Field (3270 and 5250 only)	Retrieves the object reference of the Field object of a display session.
-----------------------------------	--

Screen Data Manipulation Properties

Attrib (3270 and 5250 only)	Codes with no ASCII equivalents are converted to blanks or returned as original values.
ExtendedAttrib	Sets screen data from the host to contain an extended attribute byte.
NullToSpace	Specifies if the Null character is converted to spaces when returning the host data buffer.
SubstituteChar	Specifies what character is used to replace the non-translatable IBM symbols.
OIA (3270 and 5250 only)	Returns the OIA status line information.
SendStringTimeout	Sets or retrieves a time-out value for the PS.SendString method

See Also

[PS \(property\)](#) on page 250

Attrib (property)

3270 and 5250 display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
PS.Attrib = Bool
Bool = PS.Attrib
```

ActiveX controller application

```
Object1.Attrib = Bool
Bool = Object1.Attrib
```

Description

Defines how attribute bytes of the presentation space data are processed for the **PS.GetData** and the **Field.GetData** methods.

Return Value

Returns *Bool* as Boolean.

Value	Description
FALSE	Translate EBCDIC bytes that do not have ASCII equivalents to spaces (ASCII 20h). This is the default setting.
TRUE	Do not translate EBCDIC bytes without ASCII equivalents to spaces, instead, pass their original EBCDIC values.

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This property applies only to the PS.GetData and Field.GetData methods.

WHLLAPI Reference

WHLLAPI SetSessionParameters (function 9)

ExtendedAttrib (property)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
Bool = PS.ExtendedAttrib
PS.ExtendedAttrib = Bool
```

ActiveX controller application

```
Bool = Object1.ExtendedAttrib
Object1.ExtendedAttrib = Bool
```

Description

Determines whether to include extended attribute bytes (EABs) for the **PS.GetData** and the **Field.GetData** methods.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	Do not include any extended attribute bytes (EABs) with the presentation space data. This is the default setting.
TRUE	Include extended attribute bytes (EABs) with the presentation space data. Since there is an EAB for every character that is displayed, you must define the <i>length%</i> for GetData and SetData to be twice the size of the presentation space data.

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This property applies only to the PS.GetData and Field.GetData methods.

WHLLAPI Reference

WHLLAPI SetSessionParameters (function 9)

Field (method)

3270 and 5250 display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
Dim MyFieldObj as Field
Set MyFieldObj = PS.Field(row%, column%)
```

ActiveX controller application

```
Set MyFieldObj = Object1.Field(row%, column%)
```

Description

Retrieves the object reference for the Session's Field Object.

Parameters	Description
<i>row%</i>	Specify the row value within the desired field.
<i>column%</i>	Specify the column value within the desired field.

Return Value(s)

Returns *MyFieldObj* as a Field Object.

Value	Description
MyFieldObj	As Field (for macro).
	As Object (ActiveX Automation).
Nothing	Failed to get the specified field.
LastError Value	Description
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.

ERR_INVALID_PARAM	Either the <i>row%</i> or the <i>column%</i> value is not on the screen.
ERR_SYSTEM_ERROR	System error.
ERR_TERMINATING	Session terminating.

Prerequisite

Before executing Field on a presentation space your script or program must set **SetSharing** with TO_QUERY on the object from which PS was created. It is not necessary to repeat this action.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

```
AvivaBasic Macro:
Dim MyFieldObj as Field
```

See Also

[SetSharing \(method\)](#) on page 260

FindString (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
RowColObj = PS.FindString(string%[, [row%] [, column%]])
```

ActiveX controller application

```
RowColObj = Object1.FindString(string$[, [row%] [, column%]])
```

Description

Searches the host presentation space for a specified string. The position of the first occurrence of the specified string is returned. The search operation originates from the upper left corner of the presentation space or from *row%,col%*. The search proceeds towards the end of the presentation space. If the designated string is not located, the search stops at the end of the presentation space.

Parameters	Description
<i>string\$</i>	The string to search for in the host presentation space.
<i>row%,col%</i>	The row and column number in the presentation space from where the search forward starts.

Return Value(s)

Value	Description
RowColObj	As RowCol (for macro).
	As Object (ActiveX Automation).
Nothing	Failed to perform the search. Use the LastError() (LastError - Session Object (method)) method to obtain the reason for the failure.
LastError Value	Description
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
ERR_INVALID_PARAM	The specified row and/or column is invalid.
ERR_SYSTEM_ERROR	System error.
ERR_NOTFOUND	The string is not found

Prerequisite

Before executing FindString, your script or program must set **SetSharing** as TO_QUERY on the Session object from which PS was created. It is not necessary to repeat this action. When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

```
AvivaBasic Macro:
Dim RowColObj as RowCol
```

Remarks

Use the RowCol Object properties, Row and Column to obtain the row and column values.

WHLLAPI Reference

WHLLAPI SearchPS (function 6)

See Also

[SetSharing \(method\)](#) on page 260
[RowCol Object](#) on page 216

GetCursorLocation (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
Set RowColObj = PS.GetCursorLocation
```

ActiveX controller application

```
Set RowColObj = Object1.GetCursorLocation
```

Description

Returns the current position of the cursor location as a **RowCol** object.

Return Value(s)

Value	Description
Object	As RowCol for a AvivaBasic macro script.
	As Object for ActiveX controller application.
Nothing	An error occurred, the LastError() (LastError - Session Object (method) method) should be executed to obtain the reason for the failure
LastError value	Description
ERR_NO_SETSHARING	The prerequisite SetSharing command has not been issued by the calling program.
ERR_SYSTEM_ERROR	There has been a system error.
ERR_FAIL	Session is terminating (ActiveX Automation only)
ERR_OUTOFMEMORY	Not enough memory to create object

Prerequisites

Before executing GetCursorLocation on a presentation space your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. It is not necessary to repeat this action.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

```
AvivaBasic Macro:
Dim RowColObj as RowCol
```

Remarks

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation			VT emulation		
Model	Rows	Columns	Model	Rows	Columns	Mode	Rows	Columns
2	24	80	Normal	24	80	80 cols	24	80
3	32	80	Alternate	27	132	132 cols	27	132
4	43	80						
5	27	132						
11	62	160						

5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

WHLLAPI Reference

WHLLAPI QueryCursorLocation (function 7)

See Also

[RowCol Object](#) on page 216
[SetSharing \(method\)](#) on page 260

GetData (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
sdata$ = PS.GetData([length%, row%, col%])
```

ActiveX controller application

```
sdata$ = Object1.GetData([length%, row%, col%])
```

Description

Retrieves a portion or all of presentation space for the current session. If the parameters are not specified, then you get the entire Host Screen Data. Otherwise, you get the specified length of Host Screen Data starting from the specified cursor location.

This command is affected by the value in the following properties: **Attrib**, **ExtendedAttrib** and **NullToSpace**. See description on these properties as how data string is returned to application.

Parameters	Description
<i>row%</i>	Specify the row position where the required portion of the screen begins.

<i>column%</i>	Specify the column position where the required portion of the screen begins.
<i>length%</i>	Specify the number of characters to retrieve from the presentation space, starting at the screen position specified by <i>row%</i> and <i>column%</i> .

Return Value(s)

Returns *sdata\$* as a string:

Value	Description
Any string	The requested portion of the session's presentation space, in the form of a string.
""	Failed to retrieve the requested portion of the session's presentation space. Use the LastError() (LastError - Session Object (method) method) to obtain the reason for the failure.
LastError Value	Description
0	No error.
ERR_INVALID_PARAM	Invalid parameters.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
ERR_SYSTEM_ERROR	System error.

Prerequisites

Before executing `GetData` on a presentation space your script or program must set **SetSharing** to `TO_READ` on the Session object from which PS was created. It is not necessary to repeat this action

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

To understand how presentation space data is selected, think of using a mouse or touch screen gesture to select text in a word-processor by clicking and dragging to highlight an area. Characters are selected from *row%*, *column%* to the end of row *row%*, depending on the value of *length%*, then sequentially from the beginning to the end of subsequent lines until *length%* characters have been selected. The resulting string of characters is contained in *sdata\$*.

WHLLAPI Reference

WHLLAPI CopyPS (function 5) and WHLLAPI CopyPSToStr (function 8)

See Also

[SetSharing \(method\)](#) on page 260

MaxRowColumn (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
Set RowColObj = PS.MaxRowColumn
```

ActiveX controller application

```
Set RowColObj = Object1.MaxRowColumn
```

Description

Returns a **RowCol** object containing the maximum number of rows and columns for the current screen model.

Return Value(s)

Value	Description
Object	As RowCol for a AvivaBasic macro script.
	As Object for ActiveX controller application.
Nothing	The maximum row and column could not be obtained.
LastError value	Description
ERR_SYSTEM_ERROR	There has been a system error.
ERR_FAIL	Session is terminating (ActiveX Automation only)
ERR_OUTOFMEMORY	Not enough memory to create object

Prerequisite

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

AvivaBasic Macro:

```
Dim RowColObj as RowCol
```

Remarks

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation			VT emulation		
Model	Rows	Columns	Model	Rows	Columns	Mode	Rows	Columns
2	24	80	Normal	24	80	80 cols	24	80
3	32	80	Alternate	27	132	132 cols	27	132
4	43	80						
5	27	132						
11	62	160						

5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

See Also

- [SetSharing \(method\)](#) on page 260
- [RowCol Object](#) on page 216

NullToSpace (property)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
Bool = PS.NullToSpace
PS.NullToSpace = Bool
```

ActiveX controller application

```
Object1.NullToSpace = Bool
```

Description

Defines whether null data is converted to spaces when retrieving host data for the **PS.GetData** and the **Field.GetData** methods.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
FALSE	Does not replace NULL characters held in the return buffer with spaces.
TRUE	Replaces NULL characters in the return buffer with spaces. This is the default setting.

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This property applies only to the PS.GetData and Field.GetData methods.

OIA (property)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
PS.OIA
```

Note OIA is a predefined object for macro scripts.

ActiveX controller application

```
Set MyOIAObj = Object1.OIA
```

Description

Returns the current Operator Information Area Object.

Return Value(s)

Returns MyOIAObj as an object.

Value	Description
Object	As OIA (for macro).
	As Object (ActiveX Automation).
Nothing	Failed to obtain the OIA Data.

Prerequisite

When executing from an ActiveX controller application, *Object1* must be a PS object created using the PS command. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

WHLLAPI Reference

See WHLLAPI CopyOIA (function 13)

QueryHostUpdate (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

`rc% = PS.QueryHostUpdate`

ActiveX controller application

`rc% = Object1.QueryHostUpdate`

Description

Checks for updates to the presentation space, OIA, or both presentation space and OIA.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Constant	Value	Description
ebxHostUpdateNone	0	There have been no updates since the last QueryHostUpdate call.
ebxHostUpdateOia	1	The OIA was updated.
ebxHostUpdatePS	2	The PS was updated.
ebxHostUpdateOiaAndPS	3	Both PS and OIA were updated.
Error Messages		Click message to see value
ERR_PREREQUISITE		There was no prerequisite call, StartHostNotification
ERR_FAIL		Session is terminating (ActiveX Automation only).
Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.		

Prerequisite

A successful **StartHostNotification** call must have been made prior to calling QueryHostUpdate.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

WHLLAPI Reference

WHLLAPI QueryHostUpdate (function 24)

QueryHostUpdateWithWait (method)

3270 and 5250 display sessions only

Object

PS

Syntax

ActiveX client application

`rc% = Object1.QueryHostUpdateWithWait(timeout%)`

Description

Checks for updates to the presentation space, the OIA, or in both, and returns from the routine when the update occurs or when the time-out value expires.

Parameters	Description
timeout%	A mandatory value that specifies the amount of time, in milliseconds, that the client application will wait before continuing. This is a long variable (4 bytes) and the maximum value is 2147483647.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Constant	Value	Description
ebxHostUpdateOia	1	The OIA was updated.
ebxHostUpdatePS	2	The PS (presentation space) was updated.
ebxHostUpdateOiaAndPS	3	Both the PS and the OIA were updated.
Error Messages		
ERR_PREREQUISITE		There was no prerequisite call, StartHostNotofication.
ERR_FAIL		Session is terminating (ActiveX Automation only).
ERR_TIMEOUT		Time-out occurred while waiting for the host update on presentation space, OIA, or both.
ERR_SESSION_BUSY		Session is busy.
ERR_SYSTEM_ERROR		System error.
Refer to AvivaBasic Error Values on page 288, for a list of error values returned by AvivaBasic.		

Prerequisites

A successful **StartHostNotification** call must have been made prior to calling **QueryHostUpdateWithWait**.

When executing from an ActiveX client application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPSObj = MySession.PS
```

See Also

[QueryHostUpdate \(method\)](#) on page 193

Reset (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
PS.Reset
```

ActiveX controller application

```
object1.Reset
```

Description

Re-initializes properties of the PS Object to default values.

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

The following PS object properties are reset to their default values:

- `Attrib`
- `ExtendedAttrib`
- `NullToSpace`
- `SubstituteChar`

RetrieveKey (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
keyText$ = PS.RetrieveKey
```

ActiveX controller application

```
keyText$ = Object1.RetrieveKey
```

Description

Returns the first key in the queue which has been collected due to a prior **StartKeyIntercept** call.

Return Value(s)

Returns *keyText\$* as a string.

Value	Description
<i>keyText\$</i>	The last key pressed.
""	The queue is empty.

Prerequisite

StartKeyIntercept must have executed successfully on the session.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

Once the RetrieveKey method is executed, the returned key is removed from the queue. The key expression returned can be a normal alphanumeric key, or an AID key, depending on which option was used by the **StartKeyIntercept** call.

Return values for the RetrieveKey method are indicated in the following tables (3270 and 5250):

Note: For alphanumerics (ex."a", "1"), the characters { and } are returned as "{\}" and "{}\}".

See Also

Aviva Action Keys (function key abbreviations)

WHLLAPI Reference

WHLLAPI GetKey (function 51)

RowColToPosition (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
pos% = PS.RowColToPosition(row%, column%)
```

ActiveX controller application

```
pos% = Object1.RowColToPosition(row%, column%)
```

Description

Calculates a screen-position offset based on the specified row and column parameters.

Parameters	Description
<i>row%</i>	An integer specifying the row position of the cursor.
<i>column%</i>	An integer specifying the column position of the cursor.

Return Value(s)

Returns *pos%* as an integer.

Value	Description
0	The position offset value was converted successfully by RowColToPosition.
ERR_FAIL	Failed to perform the conversion. Use the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method to obtain the reason for the failure.
LastError value	Description
ERR_INVALID_PARAM	An invalid parameter value was used.
ERR_TERMINATING	Session terminating.
ERR_SYSTEM_ERROR	There has been a system error.

Remarks

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation			VT emulation		
Model	Rows	Columns	Model	Rows	Columns	Mode	Rows	Columns
2	24	80	Normal	24	80	80 cols	24	80
3	32	80	Alternate	27	132	132 cols	27	132
4	43	80						
5	27	132						
11	62	160						

5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

Prerequisite

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

WHLLAPI Reference

WHLLAPI Convert (function 99)

SendString (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
rc% = PS.SendString(string$, [row%] [, column%])
```

ActiveX controller application

```
rc% = Object1.SendString(string$, [row%] [, column%])
```

Description

Sends a character string, which can include 3270/5250 function and APL keys, to the host for the session associated with this PS Object. Use this command to send string and/or key from the client application to the host session. A string can be sent to any specified screen position of the host session by specifying values in the row and column parameters. When values are not specified for *row%* and *column%*, the contents of *sdata\$* are sent to the current location of the cursor.

See Remarks below for a discussion of 3270/5250 function and AID keystrokes.

Parameters	Description
<i>string\$</i>	The string to transmit to the host. If you omit the <i>row%</i> and <i>column%</i> parameters, the string is transmitted to the current cursor position. See Comments, below, for instructions on transmitting 3270/5250 AID and function keys.
<i>row%</i>	To transmit <i>string\$</i> to a specific location in the session's presentation space, you must specify an integer here. This number must not exceed the total number of rows in the session's presentation space, which depends on the type and model of the current emulation. For example, the number of rows in a model 2, 3270 display emulation is 24. Note that if you specify <i>row%</i> for this method, you must also specify <i>column%</i> .
<i>column%</i>	To transmit <i>string\$</i> to a specific screen location in the host session, you must specify an integer here. This number must not exceed the total number of columns in the session's presentation space, which depends on the type and model of the current emulation. For example, the number of columns in a model 2, 3270 display emulation is 80. Note that if you specify <i>column%</i> in the parameters for this method, you must also specify <i>row%</i> .

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	String or key sent to the host successfully.
ERR_INVALID_PARAM	Either row and/or column is out of the host screen boundaries
ERR_SESSION_INHIBITED	The string or a partial of the string was rejected by the host.
ERR_NO_SETSHARING	Application has not issued the SetSharing command.
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisites

Before executing SendString, your script or program must set **SetSharing** to SHARE_WITH_ALL on the Session object from which PS was created. It is not necessary to repeat this action.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

Function and AID keystrokes

Function and AID keystrokes are specified by bracketing specific literal strings with { and }. To make a { or } part of a string to transmit, bracket it with { and }. For example, to send "\{" you must actually send "\\{".

Only *one* function or AID keystroke can be transmitted per execution of the SendString method. After the first keystroke in *sdata*\$, SendString ignores all subsequent characters and keystrokes. To send both ordinary characters and a function or AID keystroke using the same *string*\$, you must therefore put the function or AID keystroke at the end of *sdata*\$.

Function and AID keystrokes are always transmitted to the current location of the cursor, whether or not you specify values for *row*% and *column*%.

{Reset}, {Attn} and {Sysreq} keys are always sent to the host whether the input is inhibited or not with one condition, these keys must be placed at the beginning of the key string.

Note: AID keystrokes do not apply in VT emulation.

Examples

```
rc% = PS.SendString("{Reset}abc") 'is correct
rc% = PS.SendString("abc{Reset}") 'is not correct

rc% = SendString ("abc123")      'send "abc123"
rc% = SendString ("abc123", 3, 8)'send "abc123" to row 3, column
8

rc% = SendString ("{Enter}")
rc% = SendString ("abc123{Enter}")

rc% = SendString ("Left Curly Bracket \{\{ Right Curly Bracket
\}\}")
```

Password Encryption

When recording macros, simple password encryption is supported when the user is typing in an invisible emulation field.

The macro recorder records:

```
rc% = PS.SendString("{PSW:••••}")
```

The encrypted password is replaced by special characters and is not visible to a user. When running the macro, this string is processed only if the cursor is in a field that is invisible.

WHLLAPI Reference

WHLLAPI SendKey (function 3)

See Also

[SetSharing \(method\)](#) on page 260

Aviva Action Keys (function key abbreviations) in the online Programmer's Reference

SendStringTimeout (property)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
MyTimeout& = PS.SendStringTimeout  
PS.SendStringTimeout = MyTimeout&
```

ActiveX controller application

```
MyTimeout& = Object1.SendStringTimeout  
Object1.SendStringTimeout = MyTimeout&
```

Description

Set or retrieve a time-out value (in milliseconds) for the PS.**SendString** method

Return Value

Returns *MyTimeout* as a Long value.

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyPsObj = MySession.PS
```

SetCursorPosition (method)

3270 and 5250 display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
rc% = PS.SetCursorPosition(row%, [column%])  
or  
rc% = PS.SetCursorPosition([row%], column%)
```

ActiveX controller application

```
rc% = Object1.SetCursorPosition(row%, [column%])  
or  
rc% = Object1.SetCursorPosition([row%], column%)
```

Description

Sets the cursor to the position specified by row and column values.

Parameters	Description
<i>row%</i>	Specify what the new row position of the cursor. To set only the row position of the cursor, omit the <i>column%</i> parameter.
<i>column%</i>	Specify what the new column position of the cursor. To set only the column position of the cursor, omit the <i>row%</i> parameter.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	The cursor position was successfully set.
ERR_INVALID_ROW	Invalid row.
ERR_INVALID_COLUMN	Invalid column.
ERR_NO_SETSHARING	The required SetSharing command has not been issued.
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisites

Before executing RowColToPosition on a presentation space your script or program must set **SetSharing** to SHARE_WITH_ALL on the Session object from which PS was created. It is not necessary to repeat this action.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

If row is not specified, the cursor is set to the same row with the specified column. If column is not specified, the cursor is set to the same column at the specified row.

To change only the row or column position of the cursor using this method, omit either the *row%* parameter or the *column%* parameter. If you omit the *row%* parameter a comma (,) must be used.

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation			VT emulation		
Model	Rows	Columns	Model	Rows	Columns	Mode	Rows	Columns
2	24	80	Normal	24	80	80 cols	24	80
3	32	80	Alternate	27	132	132 cols	27	132
4	43	80						
5	27	132						
11	62	160						

5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

WHLLAPI Reference

WHLLAPI SetCursor (function 40)

See Also

[MaxRowColumn \(method\)](#) on page 190

[SetSharing \(method\)](#) on page 260

SetData (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
rc% = PS.SetData(string$[, [row%] [, column%]])
```

ActiveX controller application

```
rc% = Object1.SetData(string$[, [row%] [, column%]])
```

Description

Copies the specified string to a location on the host presentation space. If row and column are not specified, the string is copied to the current cursor location.

Parameters	Description
<i>string\$</i>	A string of ASCII data to copy to the presentation space.
<i>row%</i>	Specify, (option), the row position.
<i>column%</i>	Specify, (option), the column position.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	No error.
ERR_INVALID_PARAM	Invalid parameters.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisite

Before executing SetData on a presentation space your script or program must set **SetSharing** to SHARE_WITH_ALL on the Session object from which PS was created. It is not necessary to repeat this action.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

WHLLAPI Reference

WHLLAPI CopyStrToPS (function 15)

See Also

[SetSharing \(method\)](#) on page 260

StartHostNotification (method)

Display sessions only**Object**

PS

Syntax**AvivaBasic Macro**

```
rc% = PS.StartHostNotification(type%)
```

ActiveX controller application

```
rc% = object1.StartHostNotification(type%)
```

Description

Starts the host notification process.

Parameters

Specify *type%* as follows:

Constant	Value	Description
ebxHostNotifOnOia	0	Notify when Operator Information Area is updated.
ebxHpstNotifOnPS	1	Notify when host presentation space is updated.
ebxHostNotifOnOiaAndPS	2	Notify when both host and Operator Information Area are updated.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Values	Description
0	No error.
ERR_INVALID_PARAM	Invalid <i>type%</i> parameter.
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).
Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.	

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

Use **QueryHostUpdate** method to query wherever the host or the OIA (Operator Information Area) or both has been updated.

WHLLAPI Reference

WHLLAPI StartHostNotification (function 23)

See Also

[StopHostNotification \(method\)](#) on page 207

StartKeyIntercept (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
rc% = PS.StartKeyIntercept (keyType%)
```

ActiveX controller application

```
rc% = Object1.StartKeyIntercept (keyType%)
```

Description

Intercepts keys sent to the host by other applications and puts them in an internal queue.

Parameters	Description
keyType%	An integer specifying the types of keys to intercept: 1 - AID keystrokes 2 - ALL keystrokes

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	No error.
ERR_INVALID_PARAM	An invalid <i>keytype%</i> option was specified.
ERR_SESSION_UNAVAIL	The session is unavailable(already in use), by another application.
ERR_SYSTEM_ERROR	The command failed due to a system error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisite

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

Keys sent by the current application are not intercepted. The type of keys intercepted is specified by the parameter *KeyType%*. The calling application should use the **RetrieveKey** method to return keys from the queue in a First-In-First-Out way.

WHLLAPI Reference

WHLLAPI StartKsIntercept (function 50)

See Also

[StopKeyIntercept \(method\)](#) on page 207
 Aviva Action Keys (function key abbreviations) in the online Programmer's Reference

StopHostNotification (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

`PS.StopHostNotification`

ActiveX controller application

`object1.StopHostNotification`

Description

Stops the host notification process.

Prerequisites

The **StartHostNotification** method must have been previously executed successfully. When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This command does nothing if there was no previous successful call to **StartHostNotification**.

WHLLAPI Reference

WHLLAPI StopHostNotification (function 25)

StopKeyIntercept (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

`PS.StopKeyIntercept`

ActiveX controller application

`Object1.StopKeyIntercept`

Description

Stops the application from intercepting keystrokes from the user.

Prerequisite

The **StartKeyIntercept** method must have executed successfully.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

Stop the application from intercepting Keys from user. This command will do nothing if there was no previous successful call to **StartKeyIntercept**.

WHLLAPI Reference

WHLLAPI StopHostNotification (function 53)

SubstituteChar (property)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
subs$ = PS.SubstituteChar
PS.SubstituteChar = subs$
```

ActiveX controller application

```
subs$ = Object1.SubstituteChar
Object1.SubstituteChar = subs$
```

Description

Sets the character to be used when replacing non-translatable IBM symbols for the **PS.GetData** and the **Field.GetData** methods.

Return Value(s)

Returns sub\$ as String.

Value	Description
" " (Default)	The default character for substitution is a space.
Any Character	Specify any character as desired for the substitute character.

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This property applies only to the PS.GetData and Field.GetData methods.

WHLLAPI Reference

WHLLAPI SetSessionParameters (function 9)

WaitCursorAt (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
rc% = PS.WaitCursorAt(row%[, column%] [, timeout%])
or
rc% = PS.WaitCursorAt([row%], column% [, timeout%])
```

ActiveX controller application

```
rc% = Object1.WaitCursorAt(row%[, column%] [, timeout%])
or
rc% = Object1.WaitCursorAt([row%], column% [, timeout%])
```

Description

Waits for the cursor to move to the specified row and column.

Parameters	Description
<i>row%</i>	Specify the row value. To set only the row position of the cursor, omit the <i>column%</i> parameter. A <i>row%</i> value of 0 is ignored.
<i>column%</i>	Specify the column value. To set only the column position of the cursor, omit the <i>row%</i> parameter. A <i>column%</i> value of 0 is ignored.
<i>timeout%</i>	Specify the amount of time in milliseconds the client application waits before continuing. If no time-out value is specified, the client application waits indefinitely.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Values	Description
0	Cursor has moved to the specified row and/or column location, and control has been returned to the application or macro..
ERR_TIMEOUT	The <i>timeout%</i> value has been exceeded, and the cursor has not moved to the desired position.
ERR_INVALID_PARAM	An invalid row or column number has been used.
ERR_SESSION_BUSY	The session is busy.
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisite

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This command allows the application or macro to wait for the cursor to move to the specified screen location before continuing.

This command will wait forever unless a *timeout%* parameter is specified. If *timeout%* has expired and the cursor has not moved, this command returns with ERR_TIMEOUT.

If the row value is not specified, then column control is returned to the application or macro as soon as the cursor moves from the current position.

If the column value is not specified, then row control is returned to the application or macro as soon as the cursor moves from the current position.

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation			VT emulation		
Model	Rows	Columns	Model	Rows	Columns	Mode	Rows	Columns
2	24	80	Normal	24	80	80 cols	24	80
3	32	80	Alternate	27	132	132 cols	27	132
4	43	80						
5	27	132						
11	62	160						

5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

See Also

[WaitCursorMove \(method\)](#) on page 211

WaitCursorMove (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
rc% = PS.WaitCursorMove (numrow% [, numColumn%] [, timeout%])
or
rc% = PS.WaitCursorMove ([numrow%] , numColumn% [, timeout%])
```

ActiveX controller application

```
rc% = Object1.WaitCursorMove (numrow% [, numColumn%] [, timeout%])
or
rc% = Object1.WaitCursorMove ([numrow%] , numColumn% [, timeout%])
```

Description

Waits for the cursor to move away from the current position, or by the desired number of row(s) or/and column(s).

For example:

```
WaitCursorMove (-2, 0)
wait for the cursor to move up 2 rows from current location.
```

```
WaitCursorMove (-2, -3)
wait for the cursor to move up 2 rows and left 3 columns.
```

```
WaitCursorMove (1, 8)
wait for the cursor to move down 1 row and right 8 columns.
```

Parameters	Description
<i>numrow%</i>	Specify the number of rows up (positive value) or down (negative value) from the current position that the cursor must move before control is returned to the application or macro.

<i>Numcolumn%</i>	Specify the number of columns right (positive value) or left (negative value) from the current position that the cursor must move before control is returned to the application or macro.
<i>timeout%</i>	Specify the amount of time in milliseconds the client application will wait before continuing. If no time-out value is specified, the client application waits indefinitely.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Values	Description
0	The cursor has either moved away from the current position, or has been displaced by the specified number of row(s) and column(s).
ERR_TIMEOUT	The <i>timeout%</i> value has been exceeded, and the cursor has not moved to the desired position.
ERR_SESSION_BUSY	The session is currently busy.
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisite

Before executing WaitCursorAt, your script or program must set **SetSharing** to TO_QUERY on the Session object from which PS was created. It is not necessary to repeat this action.

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This command allows the application or macro to wait for the cursor to move away from the current position, or by the specified number of row(s) or/and column(s), before continuing.

This command will wait forever unless a *timeout%* parameter is specified during the call. If the amount of time denoted by the *timeout* parameter has expired and the cursor has not moved, this command returns with ERR_TIMEOUT.

This command does not verify out-of-screen cursor movements. These movements return ERR_TIMEOUT.

If neither of the two parameters are specified, then WaitCursorMove returns control to the calling program or macro script as soon as the cursor moves from the current position.

Location in the Host presentation space is determined by position. For example, Model 2 (a screen model supported by the 3270 emulation) is a 24 x 80 matrix where position starts at 1 (upper left corner) and ends at 1920. Most emulators represent position to the user as row-column coordinates. In this example, the Model 2 presentation space starts at row 1 - column 1 (position 1) and ends at row 24 - column 80 (position 1920).

Aviva supports the following screen models:

3270 emulation			5250 emulation			VT emulation		
Model	Rows	Columns	Model	Rows	Columns	Mode	Rows	Columns
2	24	80	Normal	24	80	80 cols	24	80
3	32	80	Alternate	27	132	132 cols	27	132
4	43	80						
5	27	132						
11	62	160						

5250 display emulators support a Presentation Space of 24 rows by 80 columns. The 25th row is displayed when an error message is received from the host or when the operator presses the SysReq key. You can use this function only when row 25 is displayed.

See Also

- [WaitCursorAt \(method\)](#) on page 209
- [SetSharing \(method\)](#) on page 260

WaitForString (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

`rc% = PS.WaitForString(string$, row%, column% [,timeout%])`

ActiveX controller application

`rc% = Object1.WaitForString(string$, row%, column% [,timeout%])`

Description

Waits for the specified string to appear before control is returned to the application. This string originates from the host, not from user input.

Parameters	Description
<i>string\$</i>	A string specifying the text.
<i>row%</i>	An integer specifying the row position of the text.
<i>column%</i>	An integer specifying the column position of the text.
<i>timeout%</i>	This integer specifies the amount of time (in milliseconds) that the client application waits before continuing. Optional.

Note: To wait for a string to appear anywhere on the screen, use `row% = 0` and `column% = 0`.

Return Value(s)

Returns `rc%` as an integer. This return code can be a constant or an error message value.

Value	Description
0	String found, wait is successful.
ERR_INVALID_PARAM	Either the row and /or column is out of the host screen boundaries
ERR_TIMEOUT	Time-out is encountered while waiting for the string sent from the host at the specified cursor location.
ERR_SESSION_BUSY	Session is busy
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisite

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This command will wait forever unless a *timeout%* parameter is specified. If *timeout%* expires without locating the designated string, this command will return with ERR_TIMEOUT.

WaitHostSettle (method)

Display sessions only

Object

PS

Syntax

AvivaBasic Macro

```
rc% = PS.WaitHostSettle(settletime% [, timeout%])
```

ActiveX controller application

```
rc% = Object1.WaitHostSettle(settletime% [, timeout%])
```

Description

Waits a specified number of milliseconds for XCOM or SYSTEM messages to be cleared from the Operator Information Area (OIA) of the session associated with this PS Object.

The *settletime%* parameter is required since the host can fluctuate from ready to a busy state within a short period of time. This guarantees that the OIA is cleared for user input.

Use the optional *timeout%* parameter when the host remains unavailable for a long period of time.

Parameters	Description
<i>settletime%</i>	An integer specifying the amount of time (in milliseconds) the host remains available for user input.
<i>timeout%</i>	An integer, optionally specify the amount of time (in milliseconds) the client application will wait for the host to be available for user input.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	The session now is ready for user input.
ERR_TIMEOUT	Time-out occurred while waiting for the host to be available for user input.
ERR_SESSION_BUSY	Session is busy
ERR_SYSTEM_ERROR	System error.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisites

When executing from an ActiveX controller application, create a PS object by using the PS method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyPsObj = MySession.PS
```

Remarks

This command waits for the XCLOCK or XSYSTEM messages to be cleared from the Operator Information Area (OIA).

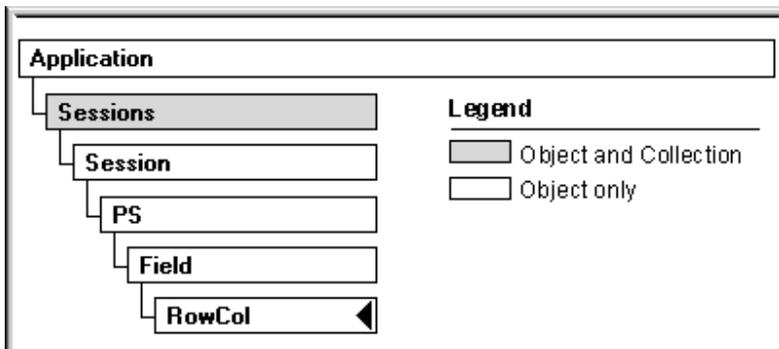
This command is called usually immediately after a SendString() call on Function/AID keys.

See Also

[SendString \(method\)](#) on page 198

RowCol Object

Display sessions only



This object contains the row and column values of a cursor location. Macro scripts and Controller applications can obtain the RowCol object reference from the PS object by using the following methods and properties:

- Position (property) Field Object
- FindString (method) PS Object
- MaxRowColumn(method) PS Object
- GetCursorLocation(method) PS Object

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined data type for Macro Scripts is "RowCol".

Properties

Row	Returns the row value of the cursor location.
Column	Returns the column value of the cursor location.

Remark

In AvivaBasic, RowCol is a data type therefore, Dim MyObject as RowCol.

See Also

[Field Object](#) on page 79

[PS Object](#) on page 180

Column (property)

Display sessions only

Object

RowCol Object

Syntax

AvivaBasic

Column% = *Object1*.Column

(where *Object1* is Dim *Object1* as RowCol)

ActiveX controller application

Column% = *Object1*.Column

Description

Returns the column value from a RowCol object created by executing one of the following: FindString, GetCursorLocation, MaxRowColumn and FieldPosition.

Return Value(s)

Returns *Column%* as Integer.

Prerequisite

When executing from an ActiveX controller application, *Object1* must be a RowCol object created using one of the following methods:

- FindString()
- MaxRowColumn()
- Position()
- GetCursorLocation()

See Also

[Row \(property\)](#) on page 217

[X \(property\)](#) on page 69

[Y \(property\)](#) on page 70

Row (property)

Display sessions only

Object

RowCol Object

Syntax

AvivaBasic

row% = *Object1*.Row

(where *Object1* is Dim *Object1* as RowCol)

ActiveX controller application

row% = *Object1*.Row

Description

Returns the row value from a RowCol object created by executing one of the following: FindString, GetCursorLocation, MaxRowColumn and FieldPosition.

Return Value(s)

Returns *row%* as Integer.

Prerequisite

When executing from an ActiveX controller application, *Object1* must be a RowCol object created using one of the following methods:

- FindString()
- MaxRowColumn()
- Position()
- GetCursorLocation()

See Also

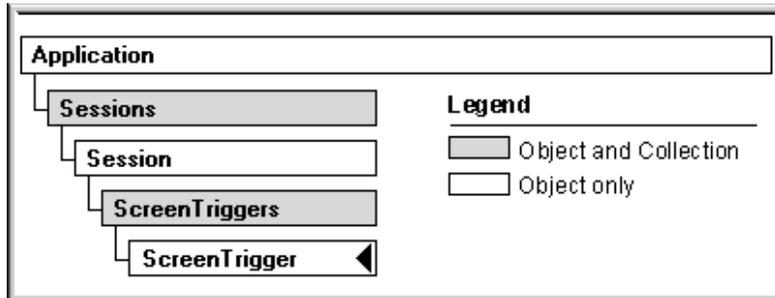
[Column \(property\)](#) on page 217

[X \(property\)](#) on page 69

[Y \(property\)](#) on page 70

ScreenTrigger Object

Display sessions only



This object provides the properties to modify Aviva screen triggers. A screen trigger is a pre-defined area of the emulation screen that Aviva searches for a specified word or phrase generated (sent to the screen) by the host and, depending on the word or phrase sent, executes a pre-assigned macro, action key, or emulation text string. The word or phrase, and the pre-defined area of the screen, are together called the “trigger condition”. Each session can have its own set of screen triggers that you customize according to your needs and preferences.

Client applications can obtain the ScreenTrigger object reference from the Session object - ScreenTriggers method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The methods and properties of this object may only be executed from an ActiveX client application

ScreenTrigger Properties

Active	Sets or retrieves the state of a ScreenTrigger.
Notify	Sets or retrieves the display user notification state of a ScreenTrigger.
MatchCase	Sets or retrieves the case sensitivity state of a ScreenTrigger.
Name	Sets or retrieves the ScreenTrigger object name.

Remarks

For ActiveX automation, the following applies:

```
Dim MySession As Object
Dim MyScreenTriggers As Object
Dim MyScreenTrigger As Object

Set MySession = GetObject("MySession.a3d")
Set MyScreenTriggers = MySession.MyScreenTriggers
Set MyScreenTrigger = MyScreenTriggers.Item
```

Active (property)

Display sessions only

Object

ScreenTrigger

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.Active = Bool  
Bool = Object1.Active
```

Description

Set or retrieve the state of a Screentrigger.

Return Value(s)

Returns *Bool* as Boolean.

Element	Description
FALSE(Default)	Not Active
TRUE	Active

Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the Session ScreenTriggers method and then create a ScreenTrigger object by using the ScreenTriggers Item method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyScreenTriggers = MySession.ScreenTriggers()  
Set MyScreenTrigger = MyScreenTriggers.item(1)
```

MatchCase (property)

Display sessions only

Object

ScreenTrigger

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.MatchCase = Bool  
Bool = Object1.MatchCase
```

Description

Set or retrieve the case sensitivity state of a ScreenTrigger. You can set any ScreenTrigger's "text to match" string as case sensitive.

Return Value(s)

Returns *Bool* as Boolean.

Element	Description
FALSE(Default)	Do not set ScreenTrigger as case sensitive.
TRUE	Set ScreenTrigger as case sensitive.

Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the Session ScreenTriggers method and then create a ScreenTrigger object by using the ScreenTriggers Item method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyScreenTriggers = MySession.ScreenTriggers()
Set MyScreenTrigger = MyScreenTriggers.item(1)
```

Name (property)**Display sessions only****Object**

ScreenTrigger

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.Name = TriggerName$
TriggerName$ = Object1.Name
```

Description

Set or retrieve the ScreenTrigger object name.

Return Value(s)

Returns *TriggerName\$* as a String.

Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the Session ScreenTriggers method and then create a ScreenTrigger object by using the ScreenTriggers Item method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyScreenTriggers = MySession.ScreenTriggers()
Set MyScreenTrigger = MyScreenTriggers.item(1)
```

Notify (property)

Display sessions only

Object

ScreenTrigger

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.Notify = Bool  
Bool = Object1.Notify
```

Description

Set or retrieve the display user notification state of a screen trigger.

Return Value(s)

Returns *Bool* as Boolean.

Element	Description
FALSE(Default)	Do not notify user.
TRUE	Notify user.

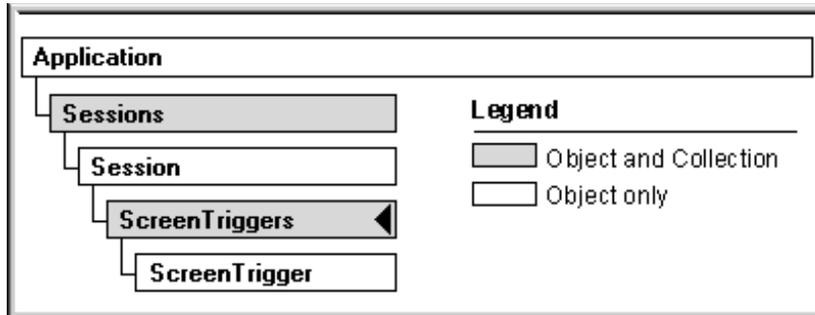
Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the Session ScreenTriggers method and then create a ScreenTrigger object by using the ScreenTriggers Item method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyScreenTriggers = MySession.ScreenTriggers()  
Set MyScreenTrigger = MyScreenTriggers.item(1)
```

ScreenTriggers Object

Display sessions only



A screen trigger is a pre-defined area of the emulation screen that Aviva searches for a specified word or phrase generated (sent to the screen) by the host and, depending on the word or phrase sent, executes a pre-assigned macro, action key, or emulation text string. The word or phrase, and the pre-defined area of the screen, are together called the "trigger condition". Each session can have its own set of screen triggers that you customize according to your needs and preferences.

A ScreenTriggers object is a collection of screen triggers. Macro scripts and client applications can obtain the ScreenTriggers object reference from the Session object - ScreenTriggers method.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. The methods and properties of this object may only be executed from an ActiveX client application

ScreenTrigger Collection Methods

Item	Retrieves a ScreenTrigger object from a collection of ScreenTriggers.
Remove	Removes a ScreenTrigger object from a collection of ScreenTriggers.
Load	Retrieves a ScreenTriggers object

ScreenTrigger Collection Properties

Count	Retrieves the number of ScreenTrigger objects from a collection of ScreenTriggers.
--------------	--

Remarks

For ActiveX automation, the following applies:

```
Dim MySession as Object
```

```
Dim MyScreenTriggers as Object
```

```
Set MySession = GetObject("MySession.a3d")
```

```
Set MyScreenTriggers = MySession.ScreenTriggers()
```

Count (property)

Display sessions only

Object

ScreenTriggers

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = Object1.Count
```

Description

Retrieve the number of ScreenTrigger objects from a collection of ScreenTriggers.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the ScreenTriggers() method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyTriggers = MySession.ScreenTriggers()
```

Item (method)

Display sessions only

Object

ScreenTriggers

Syntax

This method may only be executed from an ActiveX controller application.

```
Set Object2 = Object1.Item([index%])  
or  
Set Object2 = Object1.Item([triggername$])
```

Description

Retrieve a ScreenTrigger object from a collection of ScreenTriggers.

Parameters	Description
<i>index%</i>	The ScreenTrigger index number.
<i>triggername\$</i>	The ScreenTrigger name.

Return Value(s)

Returns *object2* as a ScreenTrigger object.

Value	Description
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_INVALID_PARAM	There is an invalid parameter.
ERR_OUTOFMEMORY	There is not enough memory to create this object.
ERR_FAIL	Cannot remove the object specified by index% or triggername\$.

Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the ScreenTriggers() method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyTriggers = MySession.ScreenTriggers()
```

Load (method)**Display sessions only****Object**

ScreenTriggers

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = Object1.Load(collectionName$, fileName$)
```

Description

Retrieve a ScreenTriggers object.

Parameters	Description
<i>collectionName\$</i>	ScreenTriggers collection name.
<i>fileName\$</i>	The name of the session or repository file where the collection of ScreenTriggers reside. If you specify a repository file, you must include a <i>collectionName\$</i> . Aviva Property Manager lets you see all the ScreenTriggers in a repository file. If you specify a session file, then set <i>collectionName\$</i> = "".

Return Value(s)

Returns *rc%* as Integer.

Value	Description
0	The command was successful.
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_SYSTEM_ERROR	The command failed due to a system error
ERR_FILENOTFOUND	The file specified by fileName\$ cannot be found
ERR_CANTOPENFILE	The file specified by fileName\$ cannot be opened
ERR_FILEACCESSERROR	There was an error accessing the file

Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the ScreenTriggers() method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyTriggers = MySession.ScreenTriggers()
```

Remarks

You can replace your current session's ScreenTriggers by loading any hotspot collection either from a session file or a repository file.

Remove (method)

Display sessions only**Object**

ScreenTriggers

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = Object1.Remove([index%])
or
rc% = Object1.Remove([triggername$])
```

Description

Remove a ScreenTrigger object from a collection of ScreenTriggers.

Parameters

Element	Description
<i>index%</i>	The ScreenTrigger index number.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

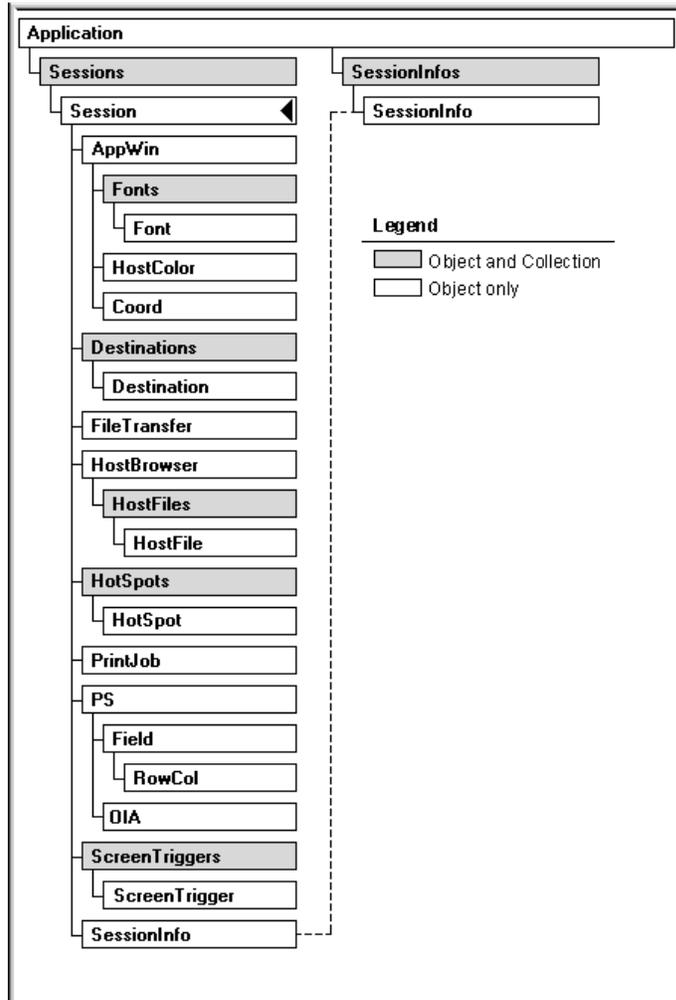
Value	Description
0	The command was successful.
Nothing	An error occurred, the LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_INVALID_PARAM	There is an invalid parameter.
ERR_OUTOFMEMORY	There is not enough memory to create this object.
ERR_FAIL	Cannot remove the object specified by <i>index%</i> or <i>triggername\$</i> .

Prerequisite

From an ActiveX client application, create a ScreenTriggers object by using the ScreenTriggers() method. For example:

```
Set MySession = GetObject("MySession.a3d")  
Set MyTriggers = MySession.ScreenTriggers()
```

Session Object



Session is the main object for all 3270, 5250, and VT host emulation sessions and the Object ID is "Aviva.Session". Client applications must first obtain the interface or reference to a session object, and then direct processing of the session object can be done.

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

Note All Printer sessions are only available through ActiveX Automation

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for macro scripts is "Session".

Macro Manipulation Methods (Display sessions only)

RunMacro	Starts the specified macro.
The following methods are ActiveX Automation only:	
BlockPlay	Disallows any execution of macros.
IsMacroRunning	Checks if any macro is currently running in a session.
UnblockPlay	Resets BlockPlay to enable macro playback.

Session Operation Methods (Display sessions only)

BlockClose	Blocks a session from closing down.
QueryBlockClose	Checks if a user has tried to close a session.
UnblockClose	Resets BlockClose, and allows a session to close.
SetProperties	Sets session properties such as session capability, host environment and screen model.
GetProperties	Retrieves the current setting of a session property.
Reset (3270 only)	Resets all Block , StartHostNotification, and StartKeyIntercept calls.
The following methods are ActiveX Automation only:	
BlockUserInput	Blocks input from user.
UnblockUserInput	Resets BlockUserInput and allow user input.

Session Operation Methods (Display and Printer sessions)

Connect	Establishes a connection between an Aviva session and the host (in a block or unblock mode).
ConnectionState	Retrieves information about the connection.
ConnectSSH	Establishes a communication link between the client and an SSH server (VT display) or an SSH Tunnel Server (3270 display and printer, 5250 display and printer)
Disconnect	Disconnects the PC session from the host.
LastError	Gets the last error encountered in the session.
Close	Closes the session.
SetSharing	Sets or resets the sharing access right.
ResetConnectParam (TN3270, TN5250, Telnet and SSH sessions only)	Reverts communication parameters to the configured values.
SetConnectParam (TN3270, TN5250, Telnet and SSH sessions only)	Sets one of the communication parameters.
The following methods are ActiveX Automation only:	
Activate	Brings a session to the foreground.

Object Creation Methods and Properties (Display sessions only)

Hotspots	Returns a Hotspots collection object or a Hotspot object.
PS	Returns a Presentation Space object.
AppWin	Returns an Application Window object.
FileTransfer (3270 only)	Returns a File Transfer object.
The following methods are ActiveX Automation only:	
Destinations (3270 only)	Returns a list of destinations configured for a session, or a specific destination from the list.
HostBrowser (3270 only)	Returns a HostBrowser object.
Screen Triggers	Returns a Screen Triggers collection object or a Screen Trigger object.

Object Creation Methods and Properties (Printer sessions only)

PrintJob	Returns a PrintJob object (ActiveX Automation only)
PrinterName	Returns the full path and name of the printer configured for use by a printer session

Properties (Display and Printer sessions)

OleTrace	Turns tracing on or off for ActiveX Automation commands (ActiveX Automation only)
SessionInfo	Retrieves information about a session

See Also

[Sessions Object](#) on page 267

Activate (method)

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

rc% = *object1*.Activate

Description

This method brings an invisible or minimized session to the foreground.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	The Activate request was successful.
ERR_FAIL	Failed to process the command.

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

See Also

[Visible \(property\)](#) on page 68

AppWin (property)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

```
Session.AppWin
```

ActiveX controller application

```
Set AppWinObj = Object1.AppWin
```

Description

Retrieves the object reference of an AppWin object. If executed from within a macro, this property is used to access the predefined object AppWin.

Return Value(s)

Returns the AppWin object reference from the display Session.

Value	Meaning
Object Reference	The object reference was successful.
NOTHING	Failed to obtain the AppWin object reference.

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

BlockClose (method)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

`rc% = Session.BlockClose`

ActiveX controller application

`rc% = object1.BlockClose`

Description

Prevents a user from closing a session.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	The BlockClose request was successful.
ERR_FAIL	Failed to process the command or a Session is terminating (ActiveX Automation only).

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

This command blocks a user from attempting to close a session either from a macro script or an Aviva menu selection. Note the following when using this command:

- The application issuing BlockClose() should reset the blocking command whenever it is not needed, or when the application is terminating.
- After a BlockClose() command, an application can check if users have tried to close the session by calling **QueryBlockClose()**.
- Also, if **SetSharing =NO_SHARING** is not reset and a BlockClose has been issued then no other controller will be able to access the PS object.

WHLLAPI Reference

WHLLAPI StartCloseIntercept (function 41)

See Also

[UnblockClose \(method\)](#) on page 263

[QueryBlockClose \(method\)](#) on page 251

BlockPlay (method)

Display sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = object1.BlockPlay
```

Description

Prevents all macros from executing, whether from user input, screen trigger, or from an ActiveX client.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	The call to block the RunMacro command was successful.
ERR_FAIL	The call to block the RunMacro command was unsuccessful.

Prerequisite

When executing from an ActiveX controller application, create a session object created by using the GetObject function or Sessions.Add. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
AvivaApp.Sessions.Add("MySession.a3d", True)
```

Remarks

Macro scripts, as well as multiple ActiveX and HLLAPI applications, can issue this call to the same session. The session keeps the corresponding usage count. This command is used to prevent any macro from executing. Execution of a macro can be initiated from user's input, screen trigger or from an ActiveX client.

See Also

- [UnblockPlay \(method\)](#) on page 264
- [RunMacro \(method\)](#) on page 254

BlockUserInput (method)

Display sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = object1.BlockUserInput
```

Description

This command is issued to prevent all user input.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	Request to prevent user input processed successfully.
ERR_SYSTEM_ERROR	Failed to process the command due to a system error.
ERR_NO_SETSHARING	Application has not issued the prerequisite SetSharing command.
ERR_SESSION_INHIBITED	Either the application does not have enough SetSharing rights to request for preventing keyboard input, or the session itself is in the input-inhibited state.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisites

Your session display object must have the right, SHARE_WITH_ALL, set by the **SetSharing** command.

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

Your application should reset BlockUserInput whenever it is not needed or before it terminates.

WHLLAPI Reference

WHLLAPI Reserve (function 11)

See Also

[UnblockUserInput \(method\)](#) on page 265

[SetSharing \(method\)](#) on page 260

Close (method)

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.Close([ForceExit%])
```

ActiveX controller application

```
rc% = object1.Close([ForceExit%])
```

Description

This command terminates the session.

Parameters	Description
<i>ForceExit%</i>	FALSE(Default) Does not terminate the session.
	TRUE Terminates the session even if it is currently in use.

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	Successful.
ERR_INVALID_PARAM	Invalid parameter.
ERR_CANNOT_EXIT_SESSION	Cannot exit session.
ERR_FAIL	Session is terminating.

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the `GetObject` function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

If *ForceExit%* is not specified or is specified as FALSE then the session will not close if the command `Session.BlockClose` has been issued to that session.

If *ForceExit%* is specified as TRUE then the session will always close.

See Also

[BlockClose \(method\)](#) on page 231
[UnblockClose \(method\)](#) on page 263

Connect (method)

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.Connect([Wait%])
```

ActiveX controller application

```
rc% = Object1.Connect([Wait%])
```

Description

Establishes a communications link between the client and host.

Parameters	Description
<i>Wait%</i>	Specify TRUE to wait until connection is successful or fail after a number of attempts (pre-configured) has been exceeded.
	Specify FALSE (default) to return immediately after an attempt to connect.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	Connection established successfully.
ERR_FAIL	Failed to make the connection (with wait flag) or Session is terminating (OLE only).
ERR_INVALID_CALL	Invalid call. There may be a disconnection in progress.
ERR_CONNECTING	Connection to host in progress.

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

When connect is issued with the blocking option, (*Wait%* flag set to TRUE), the application allows the command to either establish the connection successfully or give up trying to connect after the configured number of trials. The above return values are all possible except for ERR_CONNECTING.

If the *Wait%* flag set to FALSE, the ERR_CONNECTING code may be returned. In this case, the Communication Manager is trying to connect to the host, or going through the backup list of connection types, or re-trying the various connections. Your application can manipulate the connection state by calling **ConnectionState()**.

For macro users, a callback function, **OnConnectionFail**, is provided to notify the user of a connection failure as follows:

- The macro script has a defined a subroutine named Sub OnConnectionFail
- The macro script has previously called Session.Connect(false).
- The macro script is running when a connection failure occurs.

See Also

[Disconnect \(method\)](#) on page 240
[ConnectionState \(method\)](#) on page 238

Value	Meaning
0	Connection established successfully.
ERR_FAIL	Failed to make the connection (with wait flag) or Session is terminating (OLE only).
ERR_INVALID_CALL	Invalid call. There may be a disconnection in progress.
ERR_CONNECTING	Connection to host in progress.

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.avd")
```

Remarks

When connect is issued with the blocking option, (Wait% flag set to TRUE), the application allows the command to either establish the connection successfully or give up trying to connect after the configured number of trials. The above return values are all possible except for ERR_CONNECTING.

If the Wait% flag set to FALSE, the ERR_CONNECTING code may be returned. In this case, the Communication Manager is trying to connect to the host, or going through the backup list of connection types, or re-trying the various connections. Your application can manipulate the connection state by calling **ConnectionState()**.

For macro users, a callback function, **OnConnectionFail**, is provided to notify the user of a connection failure as follows:

- The macro script has a defined a subroutine named Sub OnConnectionFail
- The macro script has previously called Session.Connect(false).
- The macro script is running when a connection failure occurs.

See Also

[Connect \(method\) on page 235](#)

[ConnectionState \(method\) on page 238](#)

[Disconnect \(method\) on page 240](#)

ConnectionState (method)

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.ConnectionState
```

ActiveX controller application

```
rc% = object1.ConnectionState
```

Description

Determines the connection state for the active session.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Constants	Value	Meaning
ebxNotInitialized	1	Not initialized
ebxConnecting	2	Connecting
ebxDisconnecting	3	Disconnecting (Default)
ebxConnected	4	Connected
ebxDisconnected	5	Disconnected
ebxConnectionFailed	6	Connection failure.
Error Messages: Click message to see value		
ERR_SYSTEM_ERROR	Command failed due to a system error.	
ERR_FAIL	Session is terminating (ActiveX Automation only).	
Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.		

Remarks

The session must be connected before further processing can be done.

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

See Also

- [Connect \(method\)](#) on page 235
- [Disconnect \(method\)](#) on page 240
- [SetSharing \(method\)](#) on page 260

Destinations (method)

TN3270 display only

Object

Session

Syntax

This method can be executed only from an ActiveX client application.

```
Set MyDestCol = Object.Destinations([Index&])
```

Description

Returns a list of destinations configured for a session, or a specific destination from a list.

Parameters	Description
Index	As Long. Index of the destination in the collection.

Return Value(s)

Returns MyDestCol as object.

Remarks

If this method is called without a parameter, the Destination collection object is returned. If Index& (start from 1) is used as the parameter, the Destination object at the specified index is returned. When the connection state changes, the Destination collection must be recreated to contain the latest information.

Disconnect (method)

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.Disconnect([wait%])
```

ActiveX controller application

```
rc% = object1.Disconnect([wait%])
```

Description

Disconnects the communications link between the client and the host.

Parameters	Description
<i>wait%</i>	Specify TRUE to wait until connection is disconnected or fail after a number of attempts (pre-configured) has been exceeded
	Specify FALSE (default) to return immediately after an attempt to disconnect

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	Disconnected from the host successfully.
ERR_SYSTEM_ERROR	Failed to disconnect due to a system error.

ERR_DISCONNECTING	Host disconnect in progress.
ERR_INVALID_CALL	Invalid call, the session is not yet initialized.
ERR_FAIL	Session is terminating (ActiveX Automation only)

Remarks

If the *Wait%* flag set to TRUE, the application allows the command to disconnect successfully or give up trying after the configured number of trials. The above return values are all possible except for ERR_DISCONNECTING.

If the *Wait%* flag set to FALSE, the ERR_DISCONNECTING code may be returned. In this case, the Communication Manager is trying to disconnect from the host. Your application can manipulate the connection state by calling **ConnectionState()**.

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

See Also

- [Connect \(method\)](#) on page 235
- [ConnectionState \(method\)](#) on page 238

FileTransfer (property)

3270 display only

Object

Session

Syntax

AvivaBasic Macro

```
Session.FileTransfer
```

ActiveX controller application

```
Set FileTransferObj = Object1.FileTransfer
```

Description

Retrieves the object reference of a FileTransfer Object. If executed from within a macro, this property is used to access the predefined object FileTransfer.

Return Value(s)

Returns the FileTransfer Object reference from the display Session.

Value	Meaning
Object Reference	The object reference was successful.
NOTHING	Failed to obtain the FileTransfer Object reference.

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

GetProperties (method)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

```
MySessProp% = Session.GetProperties(propType%)
```

ActiveX controller application

```
MySessProp% = object1.GetProperties(propType%)
```

Description

Retrieve the current setting of a particular session property. This method only applies to Display sessions:

Parameters

propType% Specifies what property to retrieve.

Return Value(s)

Returns *MySessProp%* as an integer

For a list of property types and values, refer to the following table:

<i>propType%</i> - constants	Value	Host Environment (3270 only)
ebxPropHostEnvironment	1	Sets the session host environment
<i>MySessProp%</i> - constants	Value	Description
ebxTso	1	TSO
ebxCics	2	CICS
ebxVm,	3	VM
ebxProginet	4	PROGINET

<i>propType%</i> - constants	Value	Session capabilities (3270 only)
ebxPropCapabilities	2	Sets the session capabilities
<i>MySessProp%</i> - constants	Value	Description
ebxCapabFujitsu	1	Fujitsu Attributes
ebxCapabNumlock	2	Numeric lock

ebxCapabFieldValid	4	Field Validation
ebxCapabFieldOutline	8	Field outlining
ebxCapabAPA	16	APA graphics
ebxCapabLightPen	128	Light Pen
ebxCapabAPL	256	APL Code Page
ebxCapabS3G	32	Enable S3G and use the default symbol width and height
ebxCapabS3G9x12	64	Enable S3G and use 9 x 12 for symbol width and height

<i>propType%</i> - constants	Value	Screen Model
ebxPropScreenModel	3	Sets the session screen model
<i>MySessProp%</i> - constants	Value	Description
ebxScreen24x80	2	Model 2 (3270), Normal (5250), or 80 columns mode (VT)
ebxScreen32x80	3	Model 3 (3270 only)
ebxScreen43x80	4	Model 4 (3270 only)
ebxScreen27x132	5	Model 5 (3270), Alternate (5250)
ebxScreen24x132	6	132 columns mode (VT)
ebxScreen62x160	11	Model 11 (3270 only)

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

See Also

[SetProperties \(method\)](#) on page 258

HostBrowser (property)

Object

Session

Syntax

This method may only be executed from an ActiveX client application.

```
Set HostBrowserObj = Object1.HostBrowser
```

Description

Retrieves the object reference of a HostBrowser object.

Return Value(s)

Returns the HostBrowser Object reference from the display Session.

Value	Meaning
Object Reference	The object reference was successful.
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

The host environment must be in file transfer mode.

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

HotSpots (method)

Display sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
Set Object2 = Object1.HotSpots([Index% OR HotSpotName$])
```

Description

Returns an object reference for a Hotspots Automation Collection, or an object reference for a specified Hotspot.

Use the HotSpots method without any parameters to return the HotSpots collection - HotSpots().

Parameters	Description
index%	The HotSpot index number.
HotSpotName\$	The name of the HotSpot.

Return Value(s)

Returns *Object2* as an object.

Value	Meaning
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

IsMacroRunning (method)

Display sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
bool = object1.IsMacroRunning
```

Description

Indicates whether the specified macro is currently running.

Return Value(s)

Returns *Bool* as Boolean.

Value	Description
TRUE	There is a macro being executed.
FALSE	No macro is being executed, or the session is terminating.

Prerequisite

When executing from an ActiveX controller application, create a session object created by using the GetObject function or Sessions.Add. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
AvivaApp.Sessions.Add("MySession.a3d", True)
```

See Also

[RunMacro \(method\)](#) on page 254

LastError (method)

Object

Session

Syntax

AvivaBasic Macro

```
MyLastError% = Session.LastError
```

ActiveX controller application

```
MyLastError% = object1.LastError
```

Description

Determines the last error generated by a display or printer session.

Return Value(s)

Returns *MyLastError%* as an integer.

Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.

To retrieve the string associated with the LastError value, refer to the Application method, **LastErrorMessage**.

Prerequisite

Any command that does not return an error code.

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

Many Aviva ActiveX Automation and Macro language commands return error codes. However, some commands do not return error codes or they return a NULL string. In this case, LastError should be used to retrieve more information.

Once LastError has been called, its return value is reset to 0. It is up to the programmer to make the call at the correct time. For example, if LastError is executed at the wrong time then the error code returned may be for the previous command instead of the current one. Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.

OleTrace (property)

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
Object1.OLETrace = Bool  

Bool = Object1.OLETrace
```

Description

Sets ActiveX Automation trace on, or off.

Return Value(s)

Returns *Bool* as Boolean.

Value	Meaning
FALSE(Default)	No ActiveX Automation tracing.
TRUE	Trace on all commands to the session.

Prerequisite

When executing from an ActiveX controller application, create a Session or Printer session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

OnConnectionFail

Object

Session

Syntax

```
Sub OnConnectionFail
    'Add connection failure handler code here
End Sub
```

Description

This is not a macro command. OnConnectionFail is a callback function that notifies you of a connection failure. When this happens, the macro subsystem searches for OnConnectionFail.

However, Session.Connect(false) must be issued prior to OnConnectionFail and your macro must be running at the time of the connection failure.

Remarks

This subroutine cannot be debugged and all breakpoints will be ignored during program execution.

Prerequisite

Session.Connect(false).

See Also

[Connect \(method\)](#) on page 235

PrinterName (property)

Printer sessions only

Object

Session

Syntax

AvivaBasic Macro

```
pname$ = Session.PrinterName
```

ActiveX controller application

```
pname$ = object1.PrinterName
```

Description

Returns the full path and name of the printer configured for use by a printer session. For example, the PrinterName returns "\\MY_LAN\MY_PRINTER1".

Return Value(s)

Returns pname\$ as a String.

Value	Meaning
1-255 character string	This is the full path to the configured printer.
NULL ("")	Failed to obtain the Printer Name, or the session is terminating (ActiveX Automation only). The LastError() - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_TRY_AGAIN	There was a temporary problem in obtaining the information. Try executing PrinterName() again.
ERR_SESSIONNOTFOUND	The specified session is not active.
ERR_TERMINATING	Session is terminating.
ERR_FAIL	There was an error in executing the command.

Prerequisites

When executing from an ActiveX controller application, create a session object created by using the GetObject function or Sessions.Add. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
AvivaApp.Sessions.Add("MySession.a3p", True)
```

PrintJob (property)

Printer sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX client application.

```
Set PrintJobObj = Object1.PrintJob
```

Description

Retrieves the object reference of a PrintJob object.

Return Value(s)

Returns the PrintJob Object reference from the display Session.

Value	Meaning
Object Reference	The object reference was successful.
NOTHING	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

PS (property)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

```
Session.PS
```

ActiveX controller application

```
Set PSObj = Object1.PS
```

Description

Retrieves the object reference of a PS object. If executed from within a Macro, this property is used to access the predefined object PS.

Return Value(s)

Returns the PS object from the Display Session.

Value	Meaning
Object Reference	The object reference was successful.
NOTHING	Failed to obtain the PS object reference.

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

QueryBlockClose (method)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

`rc% = Session.QueryBlockClose`

ActiveX controller application

`rc% = object1.QueryBlockClose`

Description

Checks to see if the user has tried to close the current session after a **BlockClose()**.

Return Value(s)

Returns `rc%` as an integer. This return code can be a constant or an error message value.

Constant	Value	Meaning
<code>ebxCloseNotRequested</code>	0	The user has not made any requests to close the current session.
<code>ebxCloseRequested</code>	1	The user attempted to close the current session.
Error Messages: Click message to see value		
<code>ERR_PREREQUISITE</code>	The required prerequisite was not issued.	
<code>ERR_SYSTEM_ERROR</code>	Command failed due to a system error.	
<code>ERR_FAIL</code>	Session is terminating (ActiveX Automation only).	
Refer to the AvivaBasic error values topic for a list of error values returned by AvivaBasic.		

Prerequisites

The application must make a successful **BlockClose** request on a session.

When executing from an ActiveX controller application, create a display Session object by using the `GetObject` function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

Your application can check if a user has tried to close a session by issuing this command.

WHLLAPI Reference

WHLLAPI `QueryCloseIntercept` (function 42)

See Also

[BlockClose \(method\)](#) on page 231
[UnblockClose \(method\)](#) on page 263

Reset (method)

3270 only

Object

Session

Syntax

AvivaBasic Macro

```
Session.Reset
```

ActiveX controller application

```
object1.Reset
```

Description

This method initializes the macro and ActiveX Automation system to a default state.

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

This command is normally used at the beginning or before the end of an application to reinitialize the macro and ActiveX Automation system.

The following methods are reset:

- UnblockClose()
- UnblockUserInput()(ActiveX Automation only)
- UnblockPlay()(ActiveX Automation only)
- Stop KeyIntercept()
- StopHostNotification()

All properties of PS and FileTransfer objects are reset to default values.

ResetConnectParam (method)

TN3270, TN5250, Telnet and SSH sessions only

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.ResetConnectParam(name$)
```

ActiveX controller application

```
rc% = object1.ResetConnectParam(name$)
```

Description

Reverts communication parameters to the configured values. See [Appendix: Session.SetConnectParam Method - Parameter Values](#) on page 294 for the possible parameter names and values. The configured value will be used next time the session will connect.

Parameters

Parameter	Description
<i>name\$</i>	The name of the parameter or "*" for all parameters

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	No errors were encountered.
ERR_NOTFOUND	Invalid parameter name.

See Also

[SetConnectParam \(method\)](#) on page 257

RunMacro (method)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.RunMacro (macroName$)
```

ActiveX controller application

```
rc% = object1.RunMacro (macroName$)
```

Description

Executes a macro on the current session object.

Parameters

Element	Description
<i>macroName\$</i>	The name of the macro file. The maximum length of the file, including the full path name, is 255 characters long.

Return Value(s)

Value	Meaning
0	The macro script executed successfully.
ERR_SYSTEM_ERROR	The command failed due to a system error.
ERR_CANTOPENFILE	The file <i>macroName\$</i> could not be found.
ERR_CANNOT_COMPILE	Cannot compile the macro file
ERR_ENTRYNOTFOUND	Entry point not found in the macro file
ERR_TOOMANYMACROS	No more macros can be started because the maximum allowed cannot be exceeded
ERR_CMD_DISABLE	Macro is disabled by BlockPlay
ERR_FAIL	Session is terminating (ActiveX Automation only)

Prerequisite

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

Remarks

Macro scripts can be nested by calling RunMacro() from an other macro script or ActiveX Automation controller program. The number of nested calls allowed cannot pass the maximum number of macros which is 16.

When a macro script is executed from an other macro or ActiveX Automation controller program, execution of the calling macro or program is suspended until the called macro has run. For example, if script Macro1 executes Macro2, Macro1 only resumes

executing when Macro2 has completed or failed. However, if your macro or script makes many RunMacro calls, a called macro may not complete before the next macro starts.

See Also

- [BlockPlay \(method\)](#) on page 232
- [UnblockPlay \(method\)](#) on page 264

ScreenTriggers (method)

Display sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
Set Object2 = Object1.ScreenTriggers([Index% OR TName$])
```

Description

Returns an object reference for a Screen Triggers Automation Collection, or an object reference for a specified Screen Trigger.

Use the ScreenTriggers method without any parameters to return the ScreenTriggers collection - ScreenTriggers().

Parameters	Description
index%	The ScreenTrigger index number.
TName\$	The name of the ScreenTrigger.

Return Value(s)

Returns *Object2* as a ScreenTrigger object.

Value	Meaning
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

When executing from an ActiveX client application, create an AppWin object by using the AppWin method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set MyScreenTrig = MySession.ScreenTriggers()
```

SessionInfo (property)

Object

Session

Syntax

AvivaBasic Macro

`Session.SessionInfo`

ActiveX controller application

`Set Object2 = Object1.SessionInfo`

Description

Retrieves the object reference of a SessionInfo object. If executed from within a macro, this property is used to access the predefined object SessionInfo.

Return Value(s)

Returns *Object2* as a SessionInfo object.

Value	Meaning
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

Use this command to obtain the current session information: Short name, Long name, Emulation type and Session state. This information is returned in the SessionInfo object.

In your AvivaBasic macros, SessionInfo is a predefined object and can be accessed as SessionInfo or Session.SessionInfo.

WHLLAPI Reference

WHLLAPI ResetSystem (function 21)

See Also

[SessionInfo Object](#) on page 275

SetConnectParam (method)

TN3270, TN5250, Telnet and SSH sessions only

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.SetConnectParam(name$, value$)
```

ActiveX controller application

```
rc% = object1.SetConnectParam(name$, value$)
```

Description

Sets one of the communication parameters used by the session. See [Appendix: Session.SetConnectParam Method - Parameter Values](#) on page 294 for the possible parameter names and values. The configured value is overwritten for the lifetime of the session, but not stored in the configuration. The new value will be used next time the session will connect.

Parameters

Parameter	Description
<i>name\$</i>	The name of the parameter
<i>value\$</i>	The value of the parameter

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Description
0	No errors were encountered.
ERR_NOTFOUND	Invalid parameter name.

Note: The correctness of the *value\$* parameter is not checked by this method. The value will be checked at connection time and will cause a connection establishment failure. The cause of the failure will indicate the faulty parameter.

See Also

[ResetConnectParam \(method\)](#) on page 253

SetProperties (method)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.SetProperties(propType%,newValue%)
```

ActiveX controller application

```
rc% = object1.SetProperties(propType%,newValue%)
```

Note You can set more than one session capability by doing the following:

```
' Sets APA and Numeric lock
rc% = Session.SetProperties (2, 16 OR 2)
or
rc% = object1.SetProperties (2, 16 OR 2)
```

Description

Sets the properties of 3270, 5250, and VT display sessions.

When you set ebxPropCapabilities, you first need to call GetProperties(ebxPropCapabilities) to get the current settings, change only the bit field that represents the properties that you want to set, and then call SetProperties to apply the new setting.

Parameters

propType% Specifies the type of property to set.
newValue% The value of a property

<i>propType%</i> - constants	Value	Host Environment (3270 only)
ebxPropHostEnvironment	1	Sets the session host environment
<i>MySessProp%</i> - constants	Value	Description
ebxTso	1	TSO
ebxCics	2	CICS
ebxVm,	3	VM
ebxProginet	4	PROGINET

<i>propType%</i> - constants	Value	Session capabilities (3270 only)
ebxPropCapabilities	2	Sets the session capabilities
<i>MySessProp%</i> - constants	Value	Description
ebxCapabFujitsu	1	Fujitsu Attributes

ebxCapabNumlock	2	Numeric lock
ebxCapabFieldValid	4	Field Validation
ebxCapabFieldOutline	8	Field outlining
ebxCapabAPA	16	APA graphics
ebxCapabLightPen	128	Light Pen
ebxCapabAPL	256	APL Code Page
ebxCapabS3G	32	Enable S3G and use the default symbol width and height
ebxCapabS3G9x12	64	Enable S3G and use 9 x 12 for symbol width and height

Note: You can set either ebxCapabS3G or ebxCapabS3G9x12. You cannot set both. To change the setting, you must remove the existing setting before you apply the new one.

<i>propType%</i> - constants	Value	Screen Model
ebxPropScreenModel	3	Sets the session screen model
<i>MySessProp%</i> - constants	Value	Description
ebxScreen24x80	2	Model 2 (3270), Normal (5250), or 80 columns mode (VT)
ebxScreen32x80	3	Model 3 (3270 only)
ebxScreen43x80	4	Model 4 (3270 only)
ebxScreen27x132	5	Model 5 (3270), Alternate (5250)
ebxScreen24x132	6	132 columns mode (VT)
ebxScreen62x160	11	Model 11 (3270 only)

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Description
0	Request to enable keyboard input processed successfully.
ERR_FAIL	Session is terminating (ActiveX Automation only).
ERR_INVALID_PARAM	An invalid parameter value was used.

Prerequisite

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

See Also

[GetProperty \(method\)](#) on page 242

SetSharing (method)

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.SetSharing(shareRight%[, pwd$])
```

ActiveX controller application

```
rc% = object1.SetSharing(shareRight%[, pwd$])
```

Description

By default the first macro script or controller program to execute the SetSharing method on a session is registered with the Aviva Session Controller as the "owner" of that session. In this case *shareRight%* defines how other macro scripts or controller programs may interact with the current session.

For all subsequent macro scripts or controller programs that execute SetSharing on the same session, this method is used to request access to the session. Access is granted or denied depending on the value specified in *shareRight%* by the first macro script or controller program.

SetSharing must be release by the owner controller before exiting to prevent indefinite blocking.

In AvivaBasic you are not required to call SetSharing to perform Session methods. However, you can call SetSharing to limit access of other controllers to the current session.

Refer to Using the SetSharing Command to Control an Aviva Session for an overview of this command.

Parameters	Description
<i>shareRight%</i>	For the first script or program, <i>shareRight%</i> specifies how other macro scripts, controller programs and HLLAPI applications can or cannot access <i>object1</i> . For subsequent scripts or programs, <i>shareRight%</i> specifies how your script or program interacts with <i>object1</i> . See the table below for a list of <i>shareRight%</i> values and their descriptions.
<i>pwd\$</i>	For the first script or program, <i>pwd\$</i> specifies a password that other scripts, ActiveX controller applications, and HLLAPI applications, must supply to gain access to this session. Specify a null string ("") for this parameter if you do not require a password. Note that if you are using NO_SHARING for the <i>shareRight%</i> parameter, there is no point in using a password. For subsequent scripts or programs, <i>pwd\$</i> specifies what password is needed to gain access to the session. If the session passwords do not match, access is not granted.

shareRight% values are as follows:

Constant	ebxSharingNone
Value	6
Mnemonic	NO_SHARING
Description	Set session for exclusive read/write permission.
Restriction	Other AvivaBasic macros can access this session, but cannot issue the SetSharing command on this session. External applications have no access to this session.

Constant	ebxSharingWithQuery
Value	5
Mnemonic	SHARE_WITH_QUERY
Description	Obtain read/write permission. Allow applications set with TO_QUERY to share the same session.
Restriction	Permission is not granted if the session has been set to NO_SHARING.

Constant	ebxSharingWithReadQuery
Value	4
Mnemonic	SHARE_WITH_READ_QUERY
Description	Obtain read/write permission. Allow applications set with TO_READ or TO_QUERY to share the same session.
Restriction	Permission is not granted if the session has been set to NO_SHARING or SHARE_WITH_QUERY.

Constant	ebxSharingWithAll
Value	3
Mnemonic	SHARE_WITH_ALL
Description	Obtain read/write permission. Allow applications set with TO_QUERY, TO_READ or SHARE_WITH_ALL to share the same session.
Restriction	Permission is not granted if the session has been set to SHARE_WITH_QUERY, NO_SHARING, or SHARE_WITH_READ_QUERY.

Constant	ebxSharingToRead
Value	2

Mnemonic	TO_READ
Description	Obtain read permission to execute methods which require a SetSharing call.
Restriction	Permission is not granted if the session has already been set to NO_SHARING or SHARE_WITH_QUERY.

Constant	ebxSharingToQuery
Value	1
Mnemonic	TO_QUERY
Description	Obtain read permission.
Restriction	Permission is not granted if the session has already been set to NO_SHARING.

Constant	ebxSharingRelease
Value	0
Mnemonic	RELEASE
Description	Release a session from SetSharing access rights. This is equivalent to the WinHLLAPI DisconnectPS Function 2 .
Restriction	

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	Access to the session has been granted to your script or program and/or restrictions have been set, as requested.
ERR_SYSTEM_ERROR	SetSharing failed due to a system error.
ERR_SESSION_UNAVAIL	Session is not available or is in use.
ERR_FAIL	Session is terminating(ActiveX Automation only).

Prerequisites

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

Remarks

This is a prerequisite call to many methods provided by the Session and PS objects.

Multiple ActiveX Automation controllers, AvivaBasic macro scripts, and HLLAPI applications can request to be connected to the same session to perform various operations on the host.

An application can request for exclusive read/write permission (NO_SHARING) in which case no one else can use the same session. Also, if SetSharing = NO_SHARING is not reset and a **BlockClose (method)** has been issued then no other controller can access the object.

Other application can request for read/write permission but allow all others to perform query or read only functionality. There are 6 types of sharing rights that an application can use as detailed in the table above. The level of allowing others to share using the same session is shown in decreasing order.

The different values of *shareRight%* that can be used with this method allow multiple macro scripts or controller programs to interact with a single display session object. SetSharing must be executed before executing the following methods:

- Attribute (property)
- BlockUserInput (method)
- FindString (method)
- GetCursorLocation (method)
- GetData (method) - Field Object
- GetData (method) - PS Object
- Length (property)
- Next (method)
- NextProtected (method)
- NextUnProtected (method)
- Position (method)
- Prev (method)
- PrevProtected (method)
- PrevUnProtected (method)
- SetCursorLocation (method)
- SetData (method) - Field Object
- SetData (method) - PS Object

WHLLAPI Reference

WHLLAPI ConnectPS (function 1) and WHLLAPI DisconnectPS (function 2)

See Also

[Using the SetSharing Command to Control an Aviva Session](#) on page 16

UnblockClose (method)

Display sessions only

Object

Session

Syntax

AvivaBasic Macro

```
rc% = Session.UnblockClose
```

ActiveX controller application

```
rc% = object1.UnblockClose
```

Description

Prevents a BlockClose from terminating a session. If there are no outstanding BlockClose called by other applications, your request to close the session is processed normally.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	The request to stop intercepting user from closing the session was successful.
ERR_FAIL	Session is terminating (ActiveX Automation only).

Prerequisite

The macro script or program must issue a **BlockClose**.

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

WHLLAPI Reference

WHLLAPI StopCloseIntercept (function 43)

See Also

[BlockClose \(method\)](#) on page 231

[QueryBlockClose \(method\)](#) on page 251

UnblockPlay (method)

Display sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = object1.UnBlockPlay
```

Description

Re-enables Macro execution after a successful **BlockPlay()** call.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	Command initiated
ERR_FAIL	UnBlockPlay failed or session is terminating

Prerequisite

BlockPlay must have been issued successfully before executing UnblockPlay.

When executing from an ActiveX controller application, create an Aviva Application Object by using the CreateObject function. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
```

Remarks

If multiple programs or macro scripts have issued BlockPlay() on same session, the Session Manager automatically decrements the usage count for BlockPlay() accordingly.

See Also

[BlockPlay \(method\)](#) on page 232
[RunMacro \(method\)](#) on page 254

UnblockUserInput (method)

Display sessions only

Object

Session

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = object1.UnblockUserInput
```

Description

This command re-enables keyboard, mouse and touch screen gesture input to the host (not the menu or toolbar) after a BlockUserInput call.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Value	Meaning
0	Request to enable keyboard,mouse and touch screen gesture input processed successfully.
ERR_FAIL	Session Is terminating (ActiveX Automation only).

Prerequisites

The application must execute the BlockUserInput method.

When executing from an ActiveX controller application, create a display Session object by using the GetObject function. For example:

```
Set MySession = GetObject("MySession.a3d")
```

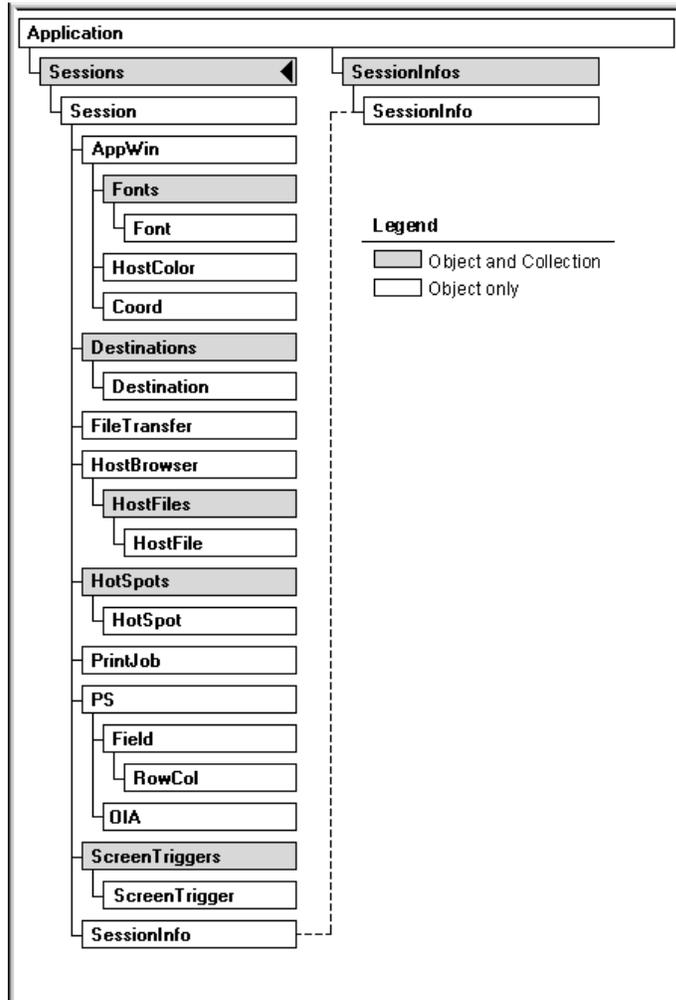
Remarks

Your application should reset BlockUserInput whenever it is not needed or before it exits.

WHLLAPI Reference

WHLLAPI Reserve (function 11)

Sessions Object



The Sessions object provides access to all active sessions as a collection. An application may use the methods provided to arrange the session windows in the workspace, return the number of active sessions, add and remove sessions and return the Object Reference to a specific session in the workspace.

Sessions Collection Methods (ActiveX Automation only)

Add	Adds the specified session to the workspace.
Arrange	Arranges the sessions: tile and cascade windows.
Item	Returns the Object Reference of the session.
Remove	Closes or decrements the usage count of the desired session. An option to force a session to quit is available.

Sessions Collection Properties (ActiveX Automation only)

ContextFilter	Gets or sets the context in which the object is operating.
Count	Returns the number of active sessions.

Add (method)

Object

Sessions

Syntax

This method can only be executed from an ActiveX client application.

```
Set object2 = Object1.Add(sessionFilename$, [Flag%])
```

Description

This method:

- Retrieves your Session from a session file.
- Adds your Session to the Sessions collection object.
- Returns your session as a Session object.
- Starts a session

Parameters		Description
<i>sessionFileName\$</i>		The name of the Session to add to the Workspace. The maximum length of the file, including the full path name, is 255 characters long.
Flag	Value	Description
True/False		This parameter is optional. Determines whether the session added is visible or not visible. If this method is called on a session that is active and the value is TRUE, then the session window is made visible. If FALSE is specified, the GUI is loaded but not shown.

ebxNoGui	2	This parameter is optional. If specified, the session is started without loading the Aviva user interface. You can combine this flag with the ebxNotInUse flag using the OR operator.
ebxNotInUse	4	This parameter is optional. If specified, the session is started and marked as not in use. You can combine this flag with the ebxNoGui flag using the OR operator.

Note Running under Aviva Windows Services, the above optional parameters are ignored and the added session always starts without Aviva's GUI.

Return Values

Returns *object2* as a Session object.

Value	Meaning
Nothing	An error occurred.
LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.	
LastError Value	Description
ERR_TOOMANYFILES	Too many sessions are already open.
ERR_TERMINATING	Session closing.
ERR_INVALID_PARAM	There is an invalid parameter
ERR_INVALID_SESSION_FILE	Invalid session file.
ERR_OLENOTAUTOMATIONOBJECT	Could not activate ActiveX Automation.
ERR_CANNOT_START_SESSION	Cannot start the session.
ERR_CANNOT_ACTIVATE_SESSION	Cannot activate the UI on the current session.

Prerequisite

When executing from an ActiveX client application, create a Sessions object by using the Sessions method. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
Set Object1 = AvivaApp.Sessions()
```

Remarks

You must execute the **Connect()** method on *Object2* to establish a session-to-host connection.

See Also

[Session Object](#) on page 228

Arrange (method)

Object

Sessions

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = Object1.Arrange([fArrange%])
```

Description

Arranges the visible Aviva session windows: tile vertically, tile horizontally, or cascade.

Parameters

Specify *fArrange%* as follows:

Constants	Value	Description
ebxTileVertical	1	Tile Vertically (default)
ebxTileHorizontal	2	Tile Horizontally
ebxCascade	3	Cascade

Return Value(s)

Returns *rc%* as an integer. This return code can be a constant or an error message value.

Value	Meaning
ERR_UNABLETOLOADDLL	Unable to load a DLL.
ERR_SYSTEM_ERROR	A system error was encountered.
ERR_INVALID_PARAM	An invalid parameter was specified.
ERR_SESSIONNOTFOUND	There are no available sessions.
0	No errors were encountered.

Prerequisite

When executing from an ActiveX controller application, create a Sessions object by using the Sessions method. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
Set Object1 = AvivaApp.Sessions()
```

ContextFilter (property)

Object

Sessions

Syntax

This property may only be executed from an ActiveX client application.

```
ContextFilterConstants = object.ContextFilter
```

```
object.ContextFilter = ContextFilterConstants
```

Description

This property is used to get or set the context in which the object is operating.

Parameters

Constants	Value	Description
ebxAllContexts	1	The collection returns all interactive sessions and sessions running as Windows services.
ebxCurrentContext	2	Default setting. This collection returns only on sessions running in the context of the caller.

Remarks

Administrator rights are required to write to this property. If the caller does not have administrator rights, this property generates a permission denied error.

Count (property)

Object

Sessions

Syntax

This property may only be executed from an ActiveX controller application.

```
num% = object1.Count
```

Description

Returns the number of sessions in the workspace.

Return Value(s)

Returns *num%* as an integer.

Value	Description
ERR_FAIL	An error occurred.

Prerequisite

When executing from an ActiveX controller application, create a Sessions object by using the Sessions method. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
Set Object1 = AvivaApp.Sessions()
```

See Also

[SessionInfo \(property\)](#) on page 256
[Sessions \(method\)](#) on page 48
[LastError \(method\)](#) on page 246, in "Session Object"
[LastError \(method\)](#) on page 41, in "Application Object"

Item (method)**Object**

Sessions

Syntax

This method may only be executed from an ActiveX controller application.

```
Set object2 = object1.Item([index%])
or
Set object2 = object1.Item([sessionFileName$])
```

Description

Returns a Session Object.

Parameters	Description
index%	The session index number
SessionFileName\$	The session file name.

Return Value(s)

Returns *object2* as a Session object.

Value	Meaning
Nothing	An error occurred.
LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.	
Last Error Value	Meaning
ERR_INVALID_PARAM	Invalid parameter, verify syntax.
ERR_INVALID_SESSION_FILE	Invalid session file.
ERR_SESSIONNOTFOUND	There are no available sessions
ERR_OLENOTAUTOMATIONOBJECT	Could not activate ActiveX Automation.
ERR_FAIL	A system error occurred.
ERR_OUTOFMEMORY	Not enough memory to create object.

Prerequisite

When executing from an ActiveX controller application, create a Sessions object by using the Sessions method. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")  
Set Object1 = AvivaApp.Sessions()
```

See Also

[Session Object](#) on page 228

Remove (method)

Object

Sessions

Syntax

This method may only be executed from an ActiveX controller application.

```
rc% = object1.Remove(sessionFileName$ [, bforceExit%])
```

Description

Removes the specified session from the workspace.

Parameters	Description	
<i>sessionFileName\$</i>	The name of the Session file to remove from the Workspace. The maximum length of the file, including the full path name, is 255 characters long	
<i>bforceExit%</i>	You may specify True if you want to force the session to be removed.	
	Value	Description
	FALSE(Default)	Don't remove session
TRUE	Remove the session.	

Return Value(s)

Value	Meaning
0	The specified session was successfully removed
Nothing	An error occurred.
LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.	
Last Error Value	Meaning
ERR_INVALID_PARAM	Invalid parameter, verify syntax.
ERR_INVALID_SESSION_FILE	Invalid session file.
ERR_SESSIONNOTFOUND	There are no available sessions
ERR_OLENOTAUTOMATIONOBJECT	Could not activate ActiveX Automation.
ERR_CANNOT_EXIT_SESSION	Cannot exit the specified session.

Prerequisite

When executing from an ActiveX controller application, create a Sessions object by using the Sessions method. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
Set SessCmds = AvivaApp.Sessions()
```

Remarks

Whether a session can successfully be removed depends on the following two conditions:

1. That there are no other controllers running in this session.
2. That the **BlockClose ()** method has not been issued prior to executing the Remove method.

If one or both conditions are not met the session is not removed. Instead, the usage count for the session is decreased accordingly. A session can be forced to exit, overriding the above two conditions by specifying True for the optional parameter *forceExit%*.

Executing **Remove()** does not destroy a display session. To destroy a display session, you must enter a command similar to this:

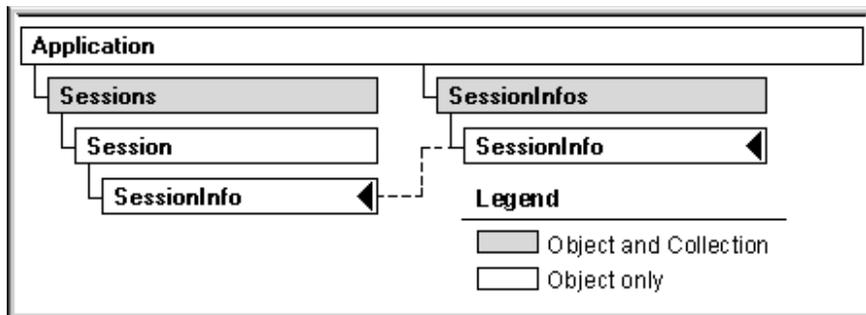
```
Set object1 = Nothing
```

Where *object1* is the name of the session file to be destroyed.

See Also

[Add \(method\)](#) on page 268
[Connect \(method\)](#) on page 235
[Disconnect \(method\)](#) on page 240
[LastError \(method\)](#) on page 246, in "Session Object"
[LastError \(method\)](#) on page 41, in "Application Object"

SessionInfo Object



This object provides information about a display or printer session. Macro scripts and controller applications can obtain the SessionInfo object reference as follows:

- Application object - SessionInfos method
- Session object - SessionInfo property

You can use ActiveX Automation to access the various objects, methods and properties of AvivaBasic. Those only available through ActiveX Automation are indicated in this document.

You can also use pre-defined macro objects and data types for AvivaBasic macro scripts. The pre-defined object name for Macro Scripts is "SessionInfo".

SessionInfo Properties

ConnectionState	Retrieves information about the session connection status. This property is available only through ActiveX Automation and only when the SessionInfo object is retrieved from the SessionInfos collection.
Context	Returns the session's security context. This property is available only through ActiveX Automation and only when the SessionInfo object is retrieved from the SessionInfos collection.
FullName	Returns the full path and name of a session.
Name	Return the Long name of a session.
Owner	Retrieves the name of the owner of the session. This property is available only through ActiveX Automation and only when the SessionInfo object is retrieved from the SessionInfos collection.
OwnerID	Retrieves the ID of the owner of the session. This property is available only through ActiveX Automation and only when the SessionInfo object is retrieved from the SessionInfos collection.
ShortName	Returns the Short name of a session.
State	Returns the state of a session.
Type	Returns the type of host emulation.

See Also

- [SessionInfos \(method\)](#) on page 47
- [SessionInfo \(property\)](#) on page 256
- [SessionInfos Object](#) on page 284

ConnectionState (property)

Object

SessionInfo

Syntax

This property can be activated only from an ActiveX client application.

```
rc% = object1.ConnectionState
```

Description

Determines the connection state of the active session.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Constants	Value	Meaning
ebxNotInitialized	1	Not initialized
ebxConnecting	2	Connecting
ebxDisconnecting	3	Disconnecting (default)
ebxConnected	4	Connected
ebxDisconnected	5	Disconnected
ebxConnectionFailed	6	Connection failure
Error messages		
ERR_SYSTEM_ERROR		Command failed due to a system error.
ERR_FAIL		Session is terminating (ActiveX Automation only).

Refer to [AvivaBasic Error Values](#) on page 288 for a list of error values returned by AvivaBasic.

Remark

The session must be connected before further processing can be done. This property is available only when the SessionInfo object is retrieved from the SessionInfos collection.

See Also

[Connect \(method\)](#) on page 235

[Disconnect \(method\)](#) on page 240

Context (property)

Object

SessionInfo

Syntax

This property can be activated only from an ActiveX client application.

`ContextConstants = object.Context`

Description

This property returns the security context of a session.

Parameters

Specify `ContextConstants` as follows:

Constants	Value	Description
ebxInteractiveContext	1	Indicates that the session is interactive.
ebxServiceContext	2	Indicates that the session runs as a Windows service.

FullName (property)

Object

SessionInfo

Syntax

AvivaBasic Macro

fname\$ = **SessionInfo**.FullName

ActiveX controller application

fname\$ = *object1*.FullName

Description

The full path and file name of a session, which can contain 1 to 255 characters. For example, the FullName returns "mypath\MySession.a3d".

Return Value(s)

Returns *fname\$* as a String.

Value	Meaning
1-255 character string	This is the full name for the current session.
NULL ("")	Failed to obtain the Full Name, or the session is terminating (ActiveX Automation only). The LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_TRY_AGAIN	There was a temporary problem in obtaining the information. Try executing Name() again.
ERR_SESSIONNOTFOUND	The specified session is not active.
ERR_TERMINATING	Session is terminating.
ERR_FAIL	There was an error in executing the command.

Prerequisites

When executing from an ActiveX controller application, create a SessionInfo object by using the SessionInfo method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set SessInfo = MySession.SessionInfo()
```

See Also

[LastError \(method\)](#) on page 246 in the Session Object
[LastError \(method\)](#) on page 41 in the Application Object

Name (property)

Object

SessionInfo

Syntax

AvivaBasic Macro

lname\$ = **SessionInfo**.Name

ActiveX controller application

lname\$ = *object1*.Name

Description

Returns the Long Name of a session object.

Return Value(s)

Returns *lname\$* as a string.

Value	Meaning
1-255 characters	This is the name of the current session.
NULL ("")	Failed to obtain the session's long name. The LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure
LastError Value	Description
ERR_TRY_AGAIN	There was a problem in obtaining the information. Try executing Name again.
ERR_SESSIONNOTFOUND	The specified session is not active.
ERR_TERMINATING	Session is terminating.
ERR_FAIL	There was an error in executing the command.

Prerequisites

When executing from an ActiveX controller application, create a SessionInfo object by using the SessionInfo property or SessionInfos method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set SessInfo = MySession.SessionInfo
or
Set AvivaApp = CreateObject("Aviva.Application")
Set SessInfo=AvivaApp.SessionInfos(1)
```

Remarks

The session Name does not contain a path or file extension. For example, the Name returned for "\mypath\MySession.a3d" is "MySession".

Owner (property)

Object

SessionInfo

Syntax`name$ = object.Owner`**Description**

This property returns the name of the owner of a session in the form DOMAIN\USER.

Returns

Returns *name\$* as a string.

Prerequisite

None

OwnerID (property)

Object

SessionInfo

Syntax`id& = object.OwnerID`**Description**

This property returns the ID of the owner of a session. By default, it has a value of 0.

Returns

Returns *id&* as an integer.

Prerequisite

None

ShortName (property)

Object

SessionInfo

Syntax**AvivaBasic Macro**`sname$ = SessionInfo.ShortName`**ActiveX controller application**`sname$ = object1.ShortName`

Description

This method obtains the API short name of the current session.

Return Value(s)

Returns *sname\$* as a String.

Value	Meaning
1 character from A-Z	This is the API short name for the current session.
NULL ("")	Failed to obtain the session's short name. The LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure
LastError Value	Description
ERR_TRY_AGAIN	There was a problem in obtaining the information. Try executing Name again.
ERR_SESSIONNOTFOUND	The specified session is not active.
ERR_TERMINATING	Session is terminating.
ERR_FAIL	There was an error in executing the command.

Remarks

A session's short name is in the range from "A" to "Z".

Prerequisites

When executing from an ActiveX controller application, create a SessionInfo object by using the SessionInfo method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set SessInfo = MySession.SessionInfo
```

See Also

[LastError \(method\)](#) on page 246 in the Session Object
[LastError \(method\)](#) on page 41 in the Application Object

State (property)

Object

SessionInfo

Syntax

AvivaBasic Macro

```
rc% = SessionInfo.State
```

ActiveX controller application

```
rc% = object1.State
```

Description

Determines the connection state for the active session.

Return Value(s)

Returns rc% as an integer. This return code can be a constant or an error message value.

Constants	Value	Meaning
ebxInitializing	0	Initializing
ebxRunning	1	Running
ebxTerminating	2	Terminating
ERR_FAIL		LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.
LastError Value		Description
ERR_SESSIONNOTFOUND		The specified session is not active.
ERR_TERMINATING		Session is terminating.
ERR_FAIL		There was an error in executing the command.

Prerequisites

When executing from an ActiveX controller application, create a SessionInfo object by using the SessionInfo method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set SessInfo = MySession.SessionInfo()
```

See Also

[Connect \(method\)](#) on page 235
[Disconnect \(method\)](#) on page 240
[SetSharing \(method\)](#) on page 260

Type (property)

Object

SessionInfo

Syntax

AvivaBasic Macro

rc% = **SessionInfo**.Type

ActiveX controller application

rc% = *object1*.Type

Description

Returns the emulation type of a session.

Return Value(s)

Returns *type%* as an integer.

Constants	Value	Meaning
ebx3270Display	1	A 3270 display session
ebx3270Printer	2	A 3270 printer session
ebx5250Display	3	A 5250 display session
ebx5250Printer	4	A 5250 printer session
ebxVTDisplay	5	A VT display session
ERR_FAIL		LastError - Session Object (method) or LastError - Application Object (method) should be executed to obtain the reason for the failure.
LastError Value		Description
ERR_SESSIONNOTFOUND		The specified session is not active.
ERR_TERMINATING		Session is terminating.

Prerequisites

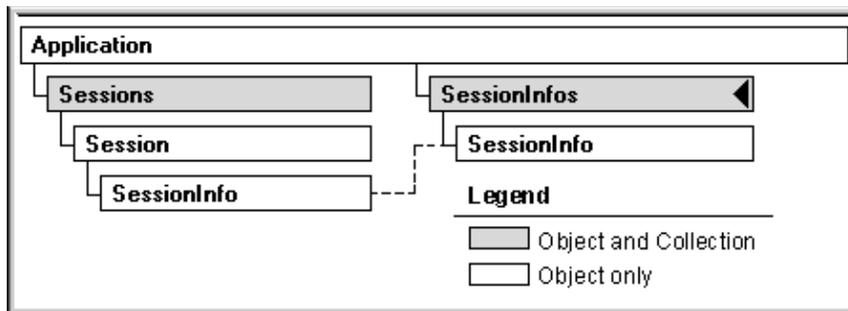
When executing from an ActiveX controller application, create a SessionInfo object by using the SessionInfo method. For example:

```
Set MySession = GetObject("MySession.a3d")
Set SessInfo = MySession.SessionInfo()
```

See Also

[LastError \(method\)](#) on page 246 in the Session Object
[LastError \(method\)](#) on page 41 in the Application Object

SessionInfos Object



The SessionInfos object provides access to an Automation Collection of SessionInfo objects, one per active session.

SessionInfos Collection Method (ActiveX Automation only)

Item	Returns the SessionInfo object reference.
-------------	---

SessionInfos Collection Property (ActiveX Automation only)

ContextFilter	Gets or sets the context in which the object is operating.
Count	Returns the number of available sessions.

ContextFilter (property)

Object

SessionInfos

Syntax

This property may only be executed from an ActiveX client application.

`ContextFilterConstants = object.ContextFilter`

`object.ContextFilter = ContextFilterConstants`

Description

This property is used to get or set the context in which the object is operating.

Parameters

Constants	Value	Description
ebxAllContexts	1	The collection returns all interactive sessions and sessions running as Windows services.
ebxCurrentContext	2	Default setting. This collection returns only on sessions running in the context of the caller.

Remarks

Administrator rights are required to write to this property. If the caller does not have administrator rights, this property generates a permission denied error.

Count (property)

Object

SessionInfos

Syntax

This property may only be executed from an ActiveX controller application.

`num% = object1.Count`

Description

Returns the number of sessions in the workspace.

Return Value

Returns `num%` as an integer.

Value	Description
ERR_FAIL	An error occurred.

Prerequisite

When executing from an ActiveX controller application, create a SessionInfo object by using the SessionInfos() method. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")
Set SessInfo = AvivaApp.SessionInfos()
```

See Also

[SessionInfos \(method\)](#) on page 47

Item (method)

Object

SessionInfos

Syntax

This method may only be executed from an ActiveX controller application.

```
Set Object2 = Object1.Item([index%])
or
Set Object2 = Object1.Item([sessionFileName$])
```

Description

Returns a SessionInfo object reference from the SessionInfos Automation collection.

Parameters	Description
index%	The session index number.
SessionFileName\$	The session file name.

Return Value(s)

Returns *object2* as a SessionInfo object.

Value	Description
Nothing	An error occurred, the LastError() (LastError - Session Object (method) or LastError - Application Object (method)) method should be executed to obtain the reason for the failure.
LastError Value	Description
ERR_INVALID_PARAM	Invalid parameter: verify your syntax is correct
ERR_INVALID_SESSION_FILE	Invalid session file.
ERR_SESSIONNOTFOUND	Specified session was not found.
ERR_OLENOTAUTOMATIONOBJECT	Could not activate ActiveX Automation.
ERR_FAIL	A system error occurred.

Prerequisite

When executing from an ActiveX controller application, create a SessionInfo object by using the SessionInfos() method. For example:

```
Set AvivaApp = CreateObject("Aviva.Application")  
Set SessInfo = AvivaApp.SessionInfos()
```

AvivaBasic Error Values

The following values are returned by AvivaBasic when an error occurs in your VBA compatible macro script.

Note These values are predefined constants in AvivaBasic. For your VBA compatible macros, define these values as constants.

Value	Name	Comment
-3	ERR_TERMINATING	This application cannot close.
-2	ERR_COMMAND	This command is not correct or has generated an error.
-1	ERR_FAIL	There has been an error.
3	ERR_RETNOGOSUB	Return without gosub
5	ERR_ILLEGAL_FUNCTION_CALL	Illegal function call
6	ERR_OVERFLOW	Overflow
7	ERR_OUTOFMEMORY	Out of memory
9	ERR_INDXOUTOFBOUNDS	Subscript out of range
11	ERR_DIVIDE0	Divide by zero
13	ERR_TYPEMISMATCH	Type mismatch
14	ERR_OUTOFSTRINGSPACE	Out of string space
19	ERR_NORESUME	No Resume
20	ERR_RESUMENOERR	Resume without error
26	ERR_DIALOG_NEEDS_BUTTON	Dialog needs End Dialog or push button
28	ERR_OUTOFSTACKSPACE	Out of stack space
35	ERR_SUBFUNCNOTDEFINED	Sub or Function not defined
48	ERR_UNABLETOLOADDLL	Error in loading DLL
49	ERR_BADDLLCALL	Bad DLL calling convention
51	ERR_INTERNAL	Internal error
52	ERR_BADFILENAME	Bad file name or number
53	ERR_FILENOTFOUND	File not found
54	ERR_BADFILEMODE	Bad file mode
55	ERR_FILEALREADYOPEN	File already open
57	ERR_DISKERROR	Device I/O error
58	ERR_FILEALREADYEXISTS	File already exists
59	ERR_BADRECORDLENGTH	Bad record length
61	ERR_DISKFULL	Disk full
62	ERR_INPUTPASTEEOF	Input past end of file
63	ERR_BADRECORDNUMBER	Bad record number
64	ERR_BADFILENAME	Bad file name
67	ERR_TOOMANYFILES	Too many files

68	ERR_INVALIDDRIVE	Device unavailable
70	ERR_ACCESSDENIED	Permission denied
71	ERR_DRIVENOTREADY	Disk not ready
74	ERR_CANTRENAME	Cannot rename with different drive
75	ERR_FILEACCESSERROR	Path/File access error
76	ERR_PATHNOTFOUND	Path not found
91	ERR_OBJECTVARIABLENOTSET	Object variable not Set
93	ERR_PATTERNINVALID	Invalid pattern string
94	ERR_INVALIDUSEOFNULL	Invalid use of Null
139	ERR_TOO_MANY_DIALOGS	Only 1 dialog may be up at any time
140	ERR_CANTFINDCONTROL	Dialog control identifier does not match any current control
141	ERR_BAD_DLG_STATEMENT	The statement is not available on this dialog control type
143	ERR_CANT_OPERATERR_FOCUS	The dialog control with the focus may not be disabled or hidden
144	ERR_CANT_SET_FOCUS	Focus may not be set to a hidden or disabled control
150	ERR_CONTROL_ALREADY_DEFINED	Dialog control identifier already defined
163	ERR_DIALOGSTATEMENTCONTEXT	This statement can only be used when a user dialog is active
260	ERR_NOTIMER	No timer available
281	ERR_TOOMANYCHANNELS	No more DDE channels
282	ERR_CANTINITIATE	No foreign application responded to a DDE initiate
283	ERR_TOOMANYRESPONSES	Multiple applications responded to a DDE initiate
285	ERR_APPNOTACCEPT	Foreign application wont perform DDE method or operation
286	ERR_DDETIMEOUT	Timeout while waiting for DDE response
287	ERR_DDEESCAPE	Escape key pressed during DDE operation
288	ERR_APPBUSY	Destination is busy
289	ERR_NODATA	Data not provided in DDE operation
290	ERR_DATAWRONGFORMAT	Data in wrong format
291	ERR_HOSTQUIT	Foreign application quit
292	ERR_CONVCHANGED	DDE conversation closed or changed
295	ERR_CANTPOST	Message queue filled; DDE message lost
298	ERR_NODDEML	DDE requires ddeml.dll
429	ERR_OLECreateFailed	ActiveX Automation server cannot create object

430	ERR_OLENotAutomationObject	Class does not support ActiveX Automation
431	ERR_OLECantLoadFile	ActiveX Automation server cannot load file
432	ERR_OLEFileOrObjectNameError	ActiveX Automation file or object name syntax error
433	ERR_OLEObjectNotExist	ActiveX Automation object does not exist
434	ERR_OLEAutoLinkFailed	Access to ActiveX Automation object denied
435	ERR_OLEInitialize	ActiveX initialization error
436	ERR_OLEUnsupportedType	ActiveX Automation method returned unsupported type
437	ERR_OLENoReturnValue	ActiveX Automation method did not return a value
438	ERR_OLENoPropOrMethod	ActiveX Automation no such property or method
439	ERR_OLETypeMismatch	ActiveX Automation argument type mismatch
440	ERR_OLEGeneric	ActiveX Automation error
443	ERR_OLENoDefaultValue	ActiveX Automation Object does not have a default value
460	ERR_INVALIDCLIPBOARDFORMAT	Invalid Clipboard format
520	ERR_CANTEMPTYCLIPBOARD	Cannot empty clipboard
521	ERR_CANTOPENCLIPBOARD	Cannot open clipboard
600	ERR_OLESetNotOnCollections	Set value not allowed on collections
601	ERR_OLEGetNotOnCollections	Get value not allowed on collections
603	ERR_ODBC_SQLALLOCENVFAILURE	ODBC - SQLAllocEnv failure
604	ERR_ODBC_SQLALLOCCONFAILURE	ODBC - SQLAllocConnect failure
608	ERR_ODBC_SQLFREECONNECT	ODBC - SQLFreeConnect error
610	ERR_ODBC_SQLALLOCSTMTFAILURE	ODBC - SQLAllocStmt failure
800	ERR_WRONGVERSION	Incorrect Windows version
801	ERR_WRONGDIMENSION	Too many dimensions
802	ERR_CANTFINDWINDOW	Cannot find window
803	ERR_BADMENUITEM	Cannot find menu item
804	ERR_2JOURNALS	Another queue is being flushed
805	ERR_BADCHILD	Cannot find control
806	ERR_INVALIDCHANNEL	Bad channel number
807	ERR_DATANOTAVAILABLE	Requested data not available
808	ERR_CANTCREATEPOPUP	Cannot create popup menu
809	ERR_CANCEL	Message box cancelled
810	ERR_COMMANDFAILED	Command failed
811	ERR_NET_ERROR	Network error

812	ERR_NET_FUNCTIONNOTSUPPORTED	Network function not supported
813	ERR_NET_BADPASSWORD	Bad password
814	ERR_NET_ACCESSDENIED	Network access denied
815	ERR_NET_BUSY	Network function busy
816	ERR_QUEOVERFLOW	Queue overflow
817	ERR_TOOMANYCONTROLS	Too many dialog controls
818	ERR_CANTFINDITEM	Cannot find listbox/combobox item
819	ERR_CONTROLDISABLED	Control is disabled
820	ERR_WINDOWDISABLED	Window is disabled
821	ERR_CANTWRITEINI	Cannot write to INI file
822	ERR_CANTREADINI	Cannot read from INI file
823	ERR_SOURCETARGETSAME	Cannot copy file onto itself
824	ERR_OLEUnknownObjectName	ActiveX Automation unknown object name
825	ERR_REDIMFIXEDARRAY	Redimension of a fixed array
826	ERR_CANTLOADEXTENSION	Cannot load and initialize extension
827	ERR_CANTFINDEXTENSION	Cannot find extension
828	ERR_UNSUPPORTED	Unsupported function or statement
829	ERR_ODBC_NODRIVER	Cannot find ODBC libraries
1000	ERR_SESSIONNOTFOUND	This session cannot be found.
1001	ERR_SESSION_BUSY	Session is busy
1002	ERR_SESSION_INHIBITED	Session is locked because input is inhibited
1003	ERR_SESSION_UNAVAIL	Session is not available (already in use)
1004	ERR_TRY_AGAIN	Resource is currently not available
1005	ERR_SYSTEM_ERROR	System reports an error
1006	ERR_NOT_SUPPORTED	This command is not supported by the current emulation DLL file
1007	ERR_DLL_NOT_LOADED	The DLL file is not loaded
1008	ERR_TIMEOUT	The command cannot wait because the time-out value has been exceeded
1009	ERR_INVALID_SESSION_FILE	This session file is not valid
1010	ERR_CANNOT_START_SESSION	Session cannot be started
1011	ERR_UNFORMATTED	The Host Presentation Space is not formatted
1012	ERR_INVALID_ROW	This row is not valid
1013	ERR_INVALID_COLUMN	This column is not valid
1014	ERR_INVALID_CALL	This call is not valid
1015	ERR_INVALID_PARAM	Parameters are not valid
1016	ERR_CONNECTING	This session is already attempting to connect

1017	ERR_DISCONNECTING	This session is already attempting to disconnect
1020	ERR_INVHOSTDATA	The Host has received data that is not valid
1021	ERR_HOSTERR	Final Host message reports an error
1022	ERR_NO_SETSHARING	SetSharing is required
1023	ERR_PREREQUISITE	The prerequisite function is not called
1024	ERR_NOTFOUND	The search item was not found
1025	ERR_CANNOT_EXIT_SESSION	Session cannot close
1026	ERR_XFER_ABORT	The File Transfer program has closed
1027	ERR_XFER_UNKNOWNHOST	File Transfer does not recognize the current Host
1028	ERR_XFER_HOSTNOTAVAILABLE	Host is not available
1029	ERR_SF_NOTSUPPORTED	Session LU does not support Structured Fields
1030	STATUS_NONE	No call in progress
1031	STATUS_ABORT_INPROGRESS	The call to the host is being stopped
1032	STATUS_INPROGRESS	There is a call in progress to the host
1033	STATUS_ABORTED	The call to the host has stopped
1034	STATUS_COMPLETE	The call to the host has finished
1035	ERR_INVALID_WORKSPACE_FILE	The workspace file is not valid
1036	ERR_CANNOT_ACTIVATE_SESSION	Cannot activate the session GUI
1037	ERR_CANNOT_COMPILE	Cannot compile the macro file
1038	ERR_ENTRYNOTFOUND	Entry point is not found in this macro file
1039	ERR_CANTOPENFILE	Cannot open this file
1040	ERR_TOOMANYMACROS	No more macros can be started because the maximum allowed cannot be exceeded
1041	ERR_CMD_DISABLE	This command is not available
1042	ERR_INVALID_ROW_COL	This row and column is not valid
1043	ECSVC_E_SESSIONINUSE	Aviva Service: The session is currently in use by a client
1044	ECSVC_E_PERMISSION_DENIED	Aviva Service: You do not have the rights to execute this action
1045	ECSVC_E_SESSIONNOTFOUND	Aviva Service: Session cannot be found in the session list
1046	ECSVC_E_CANTCREATECLIENTLIST	Aviva Service: Cannot create the list of internal clients
1048	ERR_MOREDATA	There is more data to be retrieved
1049	ERR_HOST_BROWSE_UNAVAIL	This host does not support the listing of files
1050	ERR_NO_PRINTER	The specified printer name does not exist.

3129	ERR_ODBC_INVALIDSQLSTATEMENT	Invalid SQL statement; expected DELETE, INSERT, PROCEDURE, SELECT, or UPDATE
3146	ERR_ODBC_CALLFAILED	ODBC -- call failed.
3148	ERR_ODBC_CONNECTIONFAILED	ODBC -- connection failed.
3276	ERR_ODBC_INVALIDDBID	Invalid database ID

See Also

[LastError \(method\)](#) on page 41, in “Application Object”

[LastError \(method\)](#) on page 246, in “Session Object”

[Error Handling](#) on page 25

Appendix: Session.SetConnectParam Method - Parameter Values

3270 Display Parameters

Connection

Host

Equivalent of **Host name or IP address** below For example:

```
SetConnectParam("Host", "myhost")
```

PortNumber

Equivalent of **TCP port** below. For example:

```
SetConnectParam("PortNumber", "2323")
```

DeviceName

Equivalent of **Device name** below. For example:

```
SetConnectParam("DeviceName", "MYDEVICE")
```

TerminalType

Equivalent of **Terminal type** below. For example:

```
SetConnectParam("TerminalType", "3279")
```

ExtendedAttributes

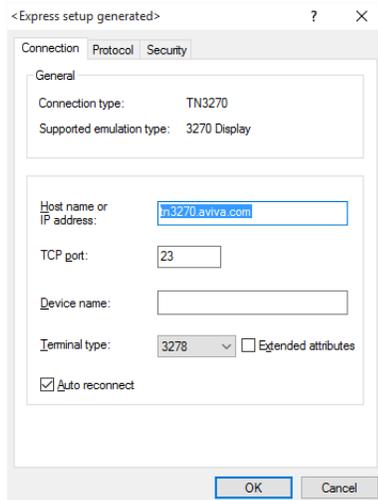
Equivalent of **Extended Attributes** below. For example:

```
SetConnectParam("ExtendedAttributes", "true")
```

AutoReconnect

Equivalent of **Auto reconnect** below. For example:

```
SetConnectParam("AutoReconnect", "false")
```



Protocol

KeepAliveType

Equivalent of **Keep-Alive/Type** below.

Values: "0"=(Disabled), "1"=NOP, "2"=TIMING-MARK

For example:

```
SetConnectParam("KeepAliveType", "1")
```

InactivityTimeout

Equivalent of **Frequency** below. For example:

```
SetConnectParam("InactivityTimeout", "900")
```

TimingMarkRspTimeout

Equivalent of **Timer** below. For example:

```
SetConnectParam("TimingMarkRspTimeout", "90")
```

Tn3270Only

Equivalent of **Use TN3270E** below.

Values: "false"=checked, "true"=unchecked For example:

```
SetConnectParam("Tn3270Only", "true")
```

ContentionResolution

Equivalent of **Contention resolution** below. For example:

```
SetConnectParam("ContentionResolution", "true")
```

SenseCodes

Equivalent of **Sense codes** below. For example:

```
SetConnectParam("SenseCodes", "true")
```

AssociatePrinterFile

Equivalent of **Associated printer file** below. For example:

```
SetConnectParam("AssociatePrinterFile",  
"C:\Users\Public\Documents\Aviva Solutions\AFD\p390.a3p")
```

CloseAssociatePrtFile

Equivalent of **Close this session when associated display session closes** below.
 For example:

```
SetConnectParam("CloseAssociatePrtFile", "true")
```



Security

EnableSecurity

Equivalent of **Enable TLS/SSL** below, along with the choice between "SSL only" and "TLS".

Values: "0"=TLS/SSL disabled, "1"=SSL only, "2"=TLS

For example:

```
SetConnectParam("EnableSecurity", "1")
```

DisplayCertificate

Equivalent of **Display certificate upon connection** below. For example:

```
SetConnectParam("DisplayCertificate", "false")
```

RejectSrvNameMismatch

Equivalent of **Reject server name mismatch** below. For example:

```
SetConnectParam("RejectSrvNameMismatch", "false")
```

EnableClientCert

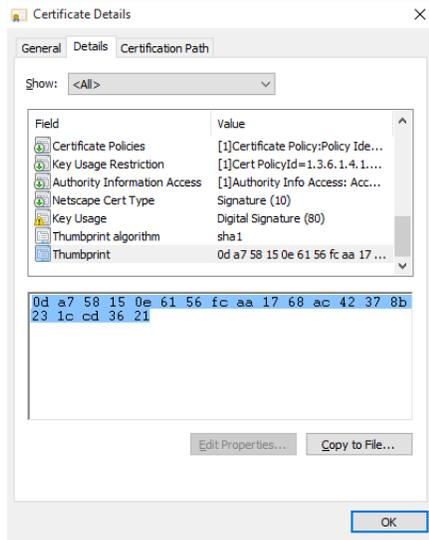
Equivalent of **Send client certificate upon connection** below. For example:

```
SetConnectParam("EnableClientCert", "true")
```

ClientCertName

Equivalent of the certificate name below.

Values: The string is a concatenation of "::- " followed by the certificate thumb print, as shown here, followed by " : " and the certificate name.



For example:

```
SetConnectParam("ClientCertName",
":: " &
"0d a7 58 15 0e 61 56 fc aa 17 68 ac 42 37 8b 23 1c cd 36 21" &
" : " &
"Aviva Solutions Inc")
```

EnableSSHTunnel

Equivalent of **Enable SSH Tunnel Server** below. For example:

```
SetConnectParam("EnableSSHTunnel", "true")
```

SSHTunnelName

Equivalent of **Enable SSH Tunnel Server/Name of IP address** below. For example:

```
SetConnectParam("SSHTunnelName", "mySSHServer")
```

SSHTunnelPort

Equivalent of **Enable SSH Tunnel Server/TCP port** below. For example:

```
SetConnectParam("SSHTunnelPort", "2222")
```

EnableProxy

Equivalent of **Enable Proxy Server** below. For example:

```
SetConnectParam("EnableProxy", "true")
```

ProxyName

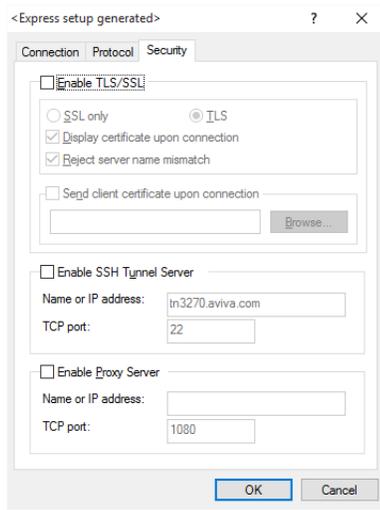
Equivalent of **Enable Proxy Server/Name of IP address** below. For example:

```
SetConnectParam("ProxyName", "myProxyServer")
```

ProxyPort

Equivalent of **Enable Proxy Server/TCP port** below. For example:

```
SetConnectParam("ProxyPort", "2080")
```



3270 Printer Parameters

Connection

Host

Equivalent of **Host name or IP address** below. For example:

```
SetConnectParam("Host", "myhost")
```

PortNumber

Equivalent of **TCP port** below. For example:

```
SetConnectParam("PortNumber", "2323")
```

DeviceName

Equivalent of **Device name** below. For example:

```
SetConnectParam("DeviceName", "MYDEVICE")
```

AssociatePrinter

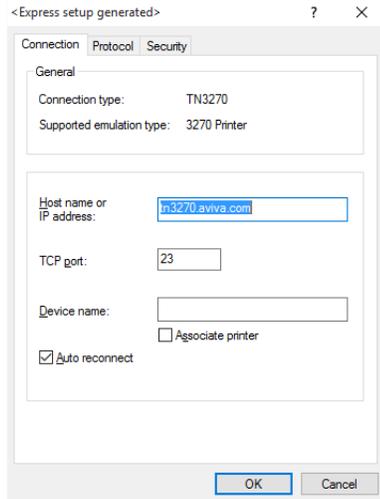
Equivalent of **Associate printer** below. For example:

```
SetConnectParam("AssociatePrinter", "true")
```

AutoReconnect

Equivalent of **Auto reconnect** below. For example:

```
SetConnectParam("AutoReconnect", "false")
```



Protocol

KeepAliveType

Equivalent of **Keep-Alive/Type** below.

Values: "0"=(Disabled), "1"=NOP, "2"=TIMING-MARK

For example:

```
SetConnectParam("KeepAliveType", "1")
```

InactivityTimeout

Equivalent of **Frequency** below. For example:

```
SetConnectParam("InactivityTimeout", "900")
```

TimingMarkRspTimeout

Equivalent of **Timer** below. For example:

```
SetConnectParam("TimingMarkRspTimeout", "90")
```

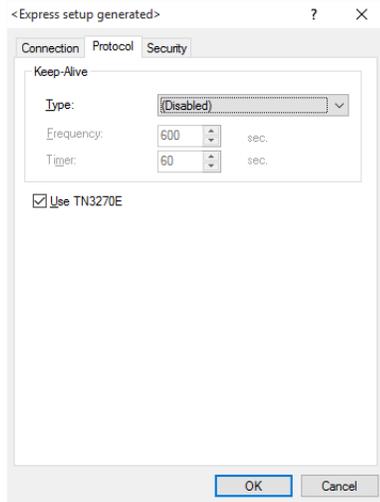
Tn3270Only

Equivalent of **Use TN3270E** below.

Values: "false"=checked, "true"=unchecked

For example:

```
SetConnectParam("Tn3270Only", "true")
```



Security

EnableSecurity

Equivalent of **Enable TLS/SSL** below, along with the choice between "SSL only" and "TLS".

Values: "0"=TLS/SSL disabled, "1"=SSL only, "2"=TLS

For example:

```
SetConnectParam("EnableSecurity", "1")
```

DisplayCertificate

Equivalent of **Display certificate upon connection** below. For example:

```
SetConnectParam("DisplayCertificate", "false")
```

RejectSrvNameMismatch

Equivalent of **Reject server name mismatch** below. For example:

```
SetConnectParam("RejectSrvNameMismatch", "false")
```

EnableClientCert

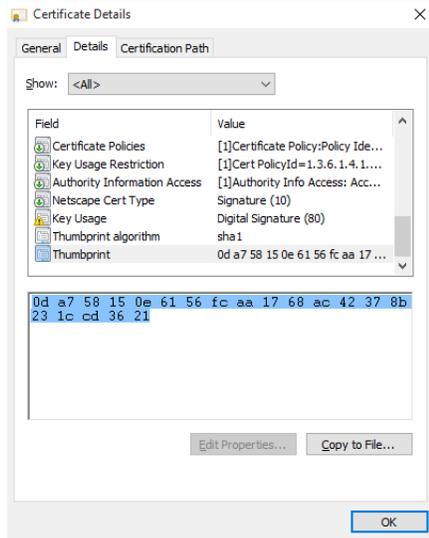
Equivalent of **Send client certificate upon connection** below. For example:

```
SetConnectParam("EnableClientCert", "true")
```

ClientCertName

Equivalent of the certificate name below.

Values: The string is a concatenation of "::- " followed by the certificate thumb print, as shown here, followed by " : " and the certificate name.



For example:

```
SetConnectParam("ClientCertName",
":: " &
"0d a7 58 15 0e 61 56 fc aa 17 68 ac 42 37 8b 23 1c cd 36 21" &
": " &
"Aviva Solutions Inc")
```

EnableSSHTunnel

Equivalent of **Enable SSH Tunnel Server** below. For example:

```
SetConnectParam("EnableSSHTunnel", "true")
```

SSHTunnelName

Equivalent of **Enable SSH Tunnel Server/Name of IP address** below. For example:

```
SetConnectParam("SSHTunnelName", "mySSHServer")
```

SSHTunnelPort

Equivalent of **Enable SSH Tunnel Server/TCP port** below. For example:

```
SetConnectParam("SSHTunnelPort", "2222")
```

EnableProxy

Equivalent of **Enable Proxy Server** below. For example:

```
SetConnectParam("EnableProxy", "true")
```

ProxyName

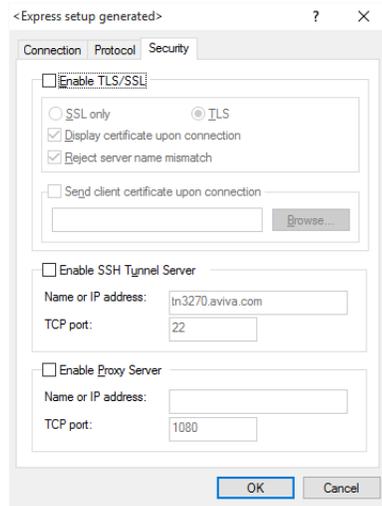
Equivalent of **Enable Proxy Server/Name of IP address** below. For example:

```
SetConnectParam("ProxyName", "myProxyServer")
```

ProxyPort

Equivalent of **Enable Proxy Server/TCP port** below. For example:

```
SetConnectParam("ProxyPort", "2080")
```



5250 Display Parameters

Connection

Host

Equivalent of **Host name or IP address** below. For example:

```
SetConnectParam("Host", "myhost")
```

PortNumber

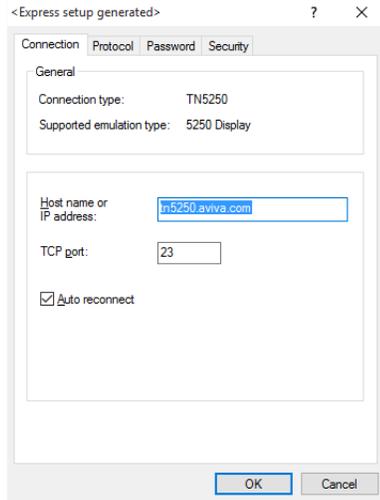
Equivalent of **TCP port** below. For example:

```
SetConnectParam("PortNumber", "2323")
```

AutoReconnect

Equivalent of **Auto reconnect** below. For example:

```
SetConnectParam("AutoReconnect", "false")
```



Protocol

KeepAliveType

Equivalent of **Keep-Alive/Type** below.

Values: "0"=(Disabled), "1"=NOP, "2"=TIMING-MARK

For example:

```
SetConnectParam("KeepAliveType", "1")
```

InactivityTimeout

Equivalent of **Frequency** below. For example:

```
SetConnectParam("InactivityTimeout", "900")
```

TimingMarkRspTimeout

Equivalent of **Timer** below. For example:

```
SetConnectParam("TimingMarkRspTimeout", "90")
```



Security

EnableSecurity

Equivalent of **Enable TLS/SSL** below, along with the choice between "SSL only" and "TLS".

Values: "0"=TLS/SSL disabled, "1"=SSL only, "2"=TLS

For example:

```
SetConnectParam("EnableSecurity", "1")
```

DisplayCertificate

Equivalent of **Display certificate upon connection** below. For example:

```
SetConnectParam("DisplayCertificate", "false")
```

RejectSrvNameMismatch

Equivalent of **Reject server name mismatch** below. For example:

```
SetConnectParam("RejectSrvNameMismatch", "false")
```

EnableClientCert

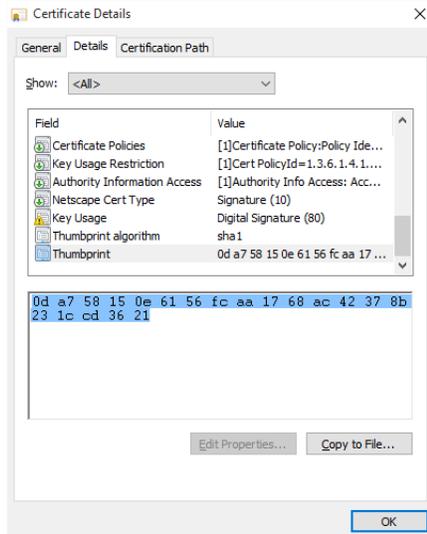
Equivalent of **Send client certificate upon connection** below. For example:

```
SetConnectParam("EnableClientCert", "true")
```

ClientCertName

Equivalent of the certificate name below.

Values: The string is a concatenation of ":: " followed by the certificate thumb print, as shown here, followed by " : " and the certificate name.



For example:

```
SetConnectParam("ClientCertName",  
": : " &  
"0d a7 58 15 0e 61 56 fc aa 17 68 ac 42 37 8b 23 1c cd 36 21" &  
" : " &  
"Aviva Solutions Inc")
```

EnableSSTunnel

Equivalent of **Enable SSH Tunnel Server** below. For example:

```
SetConnectParam("EnableSSHTunnel", "true")
```

SSHTunnelName

Equivalent of **Enable SSH Tunnel Server/Name of IP address** below. For example:

```
SetConnectParam("SSHTunnelName", "mySSHServer")
```

SSHTunnelPort

Equivalent of **Enable SSH Tunnel Server/TCP port** below. For example:

```
SetConnectParam("SSHTunnelPort", "2222")
```

EnableProxy

Equivalent of **Enable Proxy Server** below. For example:

```
SetConnectParam("EnableProxy", "true")
```

ProxyName

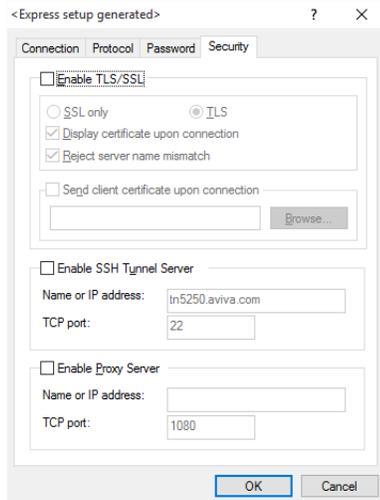
Equivalent of **Enable Proxy Server/Name of IP address** below. For example:

```
SetConnectParam("ProxyName", "myProxyServer")
```

ProxyPort

Equivalent of **Enable Proxy Server/TCP port** below. For example:

```
SetConnectParam("ProxyPort", "2080")
```



5250 Printer Parameters

Connection

Host

Equivalent of **Host name or IP address** below. For example:

```
SetConnectParam("Host", "myhost")
```

PortNumber

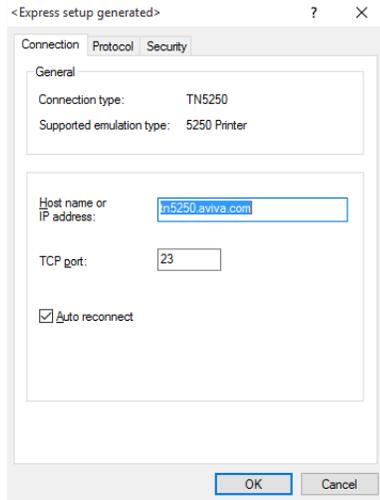
Equivalent of **TCP port** below. For example:

```
SetConnectParam("PortNumber", "2323")
```

AutoReconnect

Equivalent of **Auto reconnect** below. For example:

```
SetConnectParam("AutoReconnect", "false")
```



Protocol

KeepAliveType

Equivalent of **Keep-Alive/Type** below.

Values: "0"=(Disabled), "1"=NOP, "2"=TIMING-MARK

For example:

```
SetConnectParam("KeepAliveType", "1")
```

InactivityTimeout

Equivalent of **Frequency** below. For example:

```
SetConnectParam("InactivityTimeout", "900")
```

TimingMarkRspTimeout

Equivalent of **Timer** below. For example:

```
SetConnectParam("TimingMarkRspTimeout", "90")
```



Security

EnableSecurity

Equivalent of **Enable TLS/SSL** below, along with the choice between "SSL only" and "TLS".

Values: "0"=TLS/SSL disabled, "1"=SSL only, "2"=TLS

For example:

```
SetConnectParam("EnableSecurity", "1")
```

DisplayCertificate

Equivalent of **Display certificate upon connection** below. For example:

```
SetConnectParam("DisplayCertificate", "false")
```

RejectSrvNameMismatch

Equivalent of **Reject server name mismatch** below. For example:

```
SetConnectParam("RejectSrvNameMismatch", "false")
```

EnableClientCert

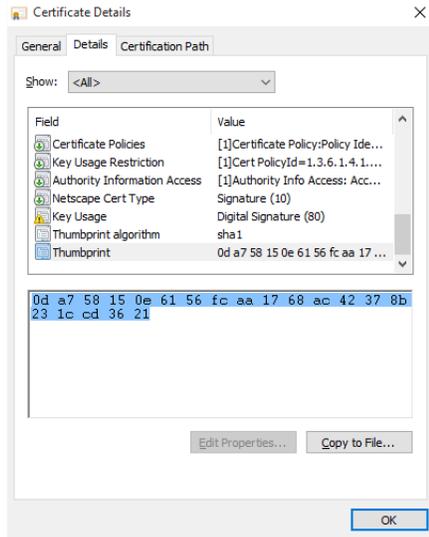
Equivalent of **Send client certificate upon connection** below. For example:

```
SetConnectParam("EnableClientCert", "true")
```

ClientCertName

Equivalent of the certificate name below.

Values: The string is a concatenation of "::- " followed by the certificate thumbprint, as shown here, followed by " : " and the certificate name.



For example:

```
SetConnectParam("ClientCertName",
":: " &
"0d a7 58 15 0e 61 56 fc aa 17 68 ac 42 37 8b 23 1c cd 36 21" &
" : " &
"Aviva Solutions Inc")
```

EnableSSHTunnel

Equivalent of **Enable SSH Tunnel Server** below. For example:

```
SetConnectParam("EnableSSHTunnel", "true")
```

SSHTunnelName

Equivalent of **Enable SSH Tunnel Server/Name of IP address** below. For example:

```
SetConnectParam("SSHTunnelName", "mySSHServer")
```

SSHTunnelPort

Equivalent of **Enable SSH Tunnel Server/TCP port** below. For example:

```
SetConnectParam("SSHTunnelPort", "2222")
```

EnableProxy

Equivalent of **Enable Proxy Server** below. For example:

```
SetConnectParam("EnableProxy", "true")
```

ProxyName

Equivalent of **Enable Proxy Server/Name of IP address** below. For example:

```
SetConnectParam("ProxyName", "myProxyServer")
```

ProxyPort

Equivalent of **Enable Proxy Server/TCP port** below. For example:

```
SetConnectParam("ProxyPort", "2080")
```



VT/Telnet Display Parameters

Connection

Host

Equivalent of **Host name or IP address** below. For example:

```
SetConnectParam("Host", "myhost")
```

PortNumber

Equivalent of **TCP port** below. For example:

```
SetConnectParam("PortNumber", "2323")
```

TerminalType

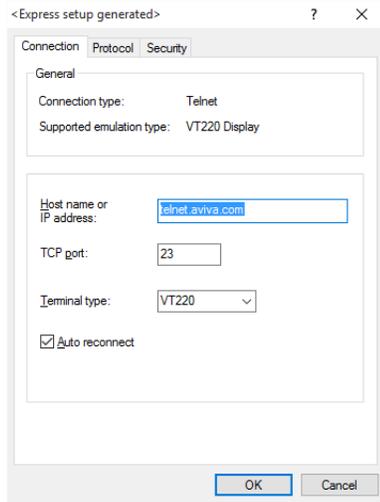
Equivalent of **Terminal type** below. For example:

```
SetConnectParam("TerminalType", "VT100")
```

AutoReconnect

Equivalent of **Auto reconnect** below. For example:

```
SetConnectParam("AutoReconnect", "false")
```



Protocol

KeepAliveType

Equivalent of **Keep-Alive/Type** below.

Values: "0"=(Disabled), "1"=NOP, "2"=TIMING-MARK

For example:

```
SetConnectParam("KeepAliveType", "1")
```

InactivityTimeout

Equivalent of **Frequency** below. For example:

```
SetConnectParam("InactivityTimeout", "900")
```

TimingMarkRspTimeout

Equivalent of **Timer** below. For example:

```
SetConnectParam("TimingMarkRspTimeout", "90")
```



Security

EnableSecurity

Equivalent of **Enable TLS/SSL** below, along with the choice between "SSL only" and "TLS".

Values: "0"=TLS/SSL disabled, "1"=SSL only, "2"=TLS

For example:

```
SetConnectParam("EnableSecurity", "1")
```

DisplayCertificate

Equivalent of **Display certificate upon connection** below. For example:

```
SetConnectParam("DisplayCertificate", "false")
```

RejectSrvNameMismatch

Equivalent of **Reject server name mismatch** below. For example:

```
SetConnectParam("RejectSrvNameMismatch", "false")
```

EnableClientCert

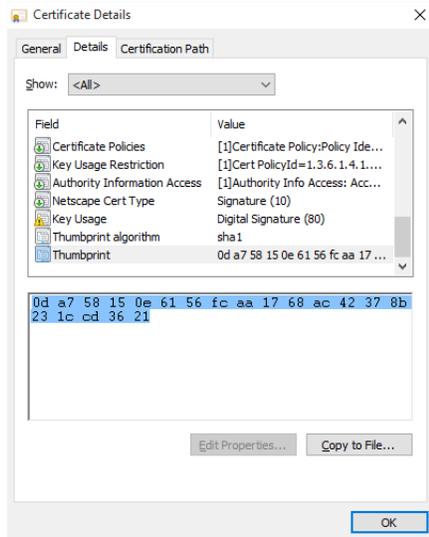
Equivalent of **Send client certificate upon connection** below. For example:

```
SetConnectParam("EnableClientCert", "true")
```

ClientCertName

Equivalent of the certificate name below.

Values: The string is a concatenation of ":: " followed by the certificate thumbprint, as shown here, followed by " : " and the certificate name.



For example:

```
SetConnectParam("ClientCertName",  
": : " &  
"0d a7 58 15 0e 61 56 fc aa 17 68 ac 42 37 8b 23 1c cd 36 21" &  
" : " &  
"Aviva Solutions Inc")
```

EnableSSTunnel

Equivalent of **Enable SSH Tunnel Server** below. For example:

```
SetConnectParam("EnableSSHTunnel", "true")
```

SSHTunnelName

Equivalent of **Enable SSH Tunnel Server/Name of IP address** below. For example:

```
SetConnectParam("SSHTunnelName", "mySSHServer")
```

SSHTunnelPort

Equivalent of **Enable SSH Tunnel Server/TCP port** below. For example:

```
SetConnectParam("SSHTunnelPort", "2222")
```

EnableProxy

Equivalent of **Enable Proxy Server** below. For example:

```
SetConnectParam("EnableProxy", "true")
```

ProxyName

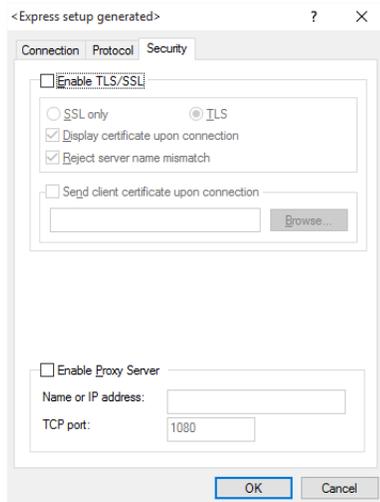
Equivalent of **Enable Proxy Server/Name of IP address** below. For example:

```
SetConnectParam("ProxyName", "myProxyServer")
```

ProxyPort

Equivalent of **Enable Proxy Server/TCP port** below. For example:

```
SetConnectParam("ProxyPort", "2080")
```



VT/SSH Display Parameters

Connection

Host

Equivalent of **Host name or IP address** below. For example:

```
SetConnectParam("Host", "myhost")
```

PortNumber

Equivalent of **TCP port** below. For example:

```
SetConnectParam("PortNumber", "2222")
```

TerminalType

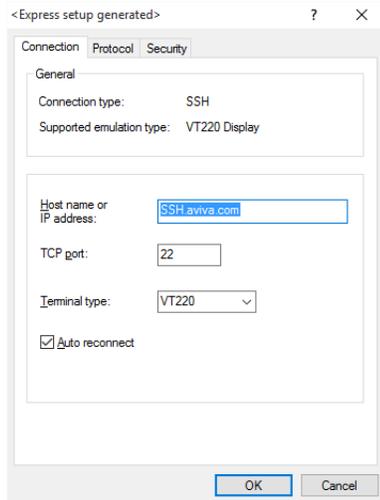
Equivalent of **Terminal type** below. For example:

```
SetConnectParam("TerminalType", "VT100")
```

AutoReconnect

Equivalent of **Auto reconnect** below. For example:

```
SetConnectParam("AutoReconnect", "false")
```



Protocol

KeepAliveType

Equivalent of **Keep-Alive/Type** below.

Values: "0"=(Disabled), "1"=NOP

For example:

```
SetConnectParam("KeepAliveType", "1")
```

InactivityTimeout

Equivalent of **Frequency** below. For example:

```
SetConnectParam("InactivityTimeout", "900")
```



Security

EnableProxy

Equivalent of **Enable Proxy Server** below. For example:

```
SetConnectParam("EnableProxy", "true")
```

ProxyName

Equivalent of **Enable Proxy Server/Name of IP address** below. For example:

```
SetConnectParam("ProxyName", "myProxyServer")
```

ProxyPort

Equivalent of **Enable Proxy Server/TCP port** below. For example:

```
SetConnectParam("ProxyPort", "2080")
```



Index

Numerics

3270 Display Parameters, 294
 3270 Printer Parameters, 298
 5250 Display Parameters, 302
 5250 Printer Parameters, 305

A

Abort (method), 94
 About EiconBasic, 12
 Activate (method), 230
 Active (property)
 HotSpot object, 156
 ScreenTrigger object, 220
 Add (method)
 Sessions object, 268
 APLMode (property), 165
 Append (property), 95
 Application object, 37–51
 Apply (method)
 Font object, 115
 HostColor object, 132
 AppWin (property), 231
 AppWin object, 52–68
 Arrange (method), 269
 Arrays, 17
 AsButton (property), 156
 AssociatePrinterFile, 295
 Attrib (property), 182
 Attribute (property)
 Field object, 80
 HostFile object, 147
 AutoReconnect, 294
 Aviva ActiveX automation overview, 12–36

B

Black (property), 133
 BlockClose (method), 231
 BlockPlay (method), 232
 BlockSize (property), 96
 BlockUserInput (method), 233
 Blue (property), 134
 Brown (property), 134
 Busy SNA Hosts, 16
 BytesTransferred (method), 96

C

Cancel (method), 174
 ClientCertName, 296
 Close (method), 234

CloseAll (method), 38
 CloseAssociatePrtFile , 295
 Column (property), 217
 CommCheck (property), 166
 Comments, 19
 Comparison operators, 19
 Connect (method), 235
 ConnectionState (method), 238
 ConnectionState (property), 276
 ConnectionType (property), 73
 Constants, 21
 ContentionResolution, 295
 Context (property), 277
 ContextFilter (property)
 SessionInfos object, 285
 Sessions object, 271
 Coordinate object, 69–71
 Copy (method), 53
 Count (property)
 Destinations object, 77
 Fonts object, 119
 HostFiles object, 152
 HotSpots object, 160
 ScreenTriggers object, 224
 SessionInfos object, 285
 Sessions object, 271
 CrLf (property), 97
 Current (method), 120

D

DarkGray (property), 135
 DecreaseFontSize (method), 54
 DeepBlue (property), 136
 Deselect (method), 55
 Dest32TN, 73
 DestGeneric Interface, 72
 Destination object, 72–??
 DestinationInUse (property), 78
 DestinationName (property), 74
 Destinations (method), 239
 Destinations object, 77–78
 DeviceName, 294
 DeviceName (property), 74
 Disconnect (method), 240
 DisplayCertificate, 296
 DisplayMode (property), 55

E

EiconBasic error values, 288–293
 EiconBasic predefined automation objects and data types, 13

EiconConfigDir (method), 39
 EiconSystemDir (method), 40
 EnableClientCert, 296
 EnableSecurity, 296
 EnableSSHTunnel, 297
 Error Handling, 25
 ExecutionDir (method), 40
 Expression Evaluation, 26
 ExtendedAttrib (property), 183
 ExtendedAttributes, 294

F

Field (method), 184
 Field object, 79–92
 FileTransfer (property), 241
 FileTransfer object, 93–113
 FindString (method), 185
 Font object, 114–117
 Fonts (method), 56
 Fonts object, 118–121
 FullName (property), 278

G

GetCursorLocation (method), 186
 GetData (method)
 Field object, 81
 PS object, 188
 GetPosition (method), 57
 GetProperties (method), 242
 GetSize (method), 58
 GraphicCursorMode (property), 167
 Gray (property), 136
 Green (property), 137

H

Host, 294
 HostBrowser (property), 243
 HostBrowser object, 122–129
 HostColor (property), 59
 HostColor object, 130–145
 HostFile (property), 98
 HostFile object, 146–150
 HostFiles (method), 124
 HostFiles object, 151–154
 HostName (property), 75
 HotSpot object, 155–158
 HotSpots (method), 245
 HotSpots object, 159–163

I

InactivityTimeout, 295
 IncreaseFontSize (method), 60
 InputInhibit (property), 167
 InputInhibitState (property), 168
 InsertMode (property), 169
 Interface

 Dest32TN, 73
 DestGeneric, 72
 IsMacroRunning (method), 245
 IsProtected (property), 82
 Item (method)
 Destinations object, 78
 Fonts object, 121
 HostFiles object, 153
 HotSpots object, 160
 ScreenTriggers object, 224
 SessionInfos object, 286
 Sessions object, 272

K

KeepAliveType, 295
 Keywords, 28

L

LastError (method)
 Application object, 41
 Session object, 246
 LastErrorMessage (method), 42
 Length (property), 83
 Line Numbers, 29
 Literals, 29
 Load (method)
 HotSpots object, 161
 ScreenTriggers object, 225
 LogFile (method), 43
 LogRecLen (property), 99

M

MachineCheck (property), 170
 MacroDir (method), 43
 Mask (property), 125
 MatchCase (property)
 Hotspot object, 157
 ScreenTrigger object, 220
 MaxCount (property), 126
 MaxRowColumn (method), 190
 MessageWaiting (property), 170
 Multiple controllers interacting with one session, 16

N

Name (property)
 Font object, 116
 HostFile object, 148
 HotSpot object, 158
 ScreenTrigger object, 221
 SessionInfo object, 279
 Named Parameters, 30
 Next (method), 84
 NextProtected (method), 85
 NextUnprotected (method), 86
 Notify (property), 222
 NullToSpace (property), 191

O

Objects, 31
 OIA (property), 192
 OIA object, 164–173
 OleTrace (property), 247
 OnConnectionFail, 248
 OnFileTransferDone, 99
 OpenWorkSpaceFile (method), 44
 Operator Precedence, 33
 Operator Precision, 34
 Orange (property), 138
 Owner (property), 280
 OwnerID (property), 280
 Ownership (property), 171

P

PA1 (method), 175
 PA2 (method), 176
 PacketSize (property), 100
 PaleGreen (property), 138
 PaleTurquoise (property), 139
 Paste (method), 60
 Pause (method), 177
 PCFile (property), 101
 Pink (property), 140
 PortNumber, 294
 PortNumber (property), 75
 Position (property), 87
 Predefined Data Types, 14
 Prev (method), 88
 PrevProtected (method), 89
 PrevUnProtected (method), 90
 PrinterName (property), 248
 PrintFile (function), 149
 PrintJob (property), 249
 PrintJob object, 174–179
 PrintScreen (method), 61
 ProgramCheck (property), 172
 PS (property), 250
 PS object, 180–215
 Purple (property), 140

Q

QueryBlockClose (method), 251
 QueryHostUpdate (method), 193
 QueryHostUpdateWithWait (method), 194

R

Receive (method), 102
 RecFormat (property), 103
 Red (property), 141
 RejectSrvNameMismatch, 296
 Remove (method)
 HotSpots object, 162
 ScreenTriggers object, 226

 Sessions object, 273
 Reset (method)
 FileTransfer object, 104
 HostBrowser object, 127
 PS object, 195
 Session object, 252
 ResetConnectParam (method), 253
 Resume (method), 178
 RetrieveKey (method), 196
 RGB (function), 142
 Row (property), 217
 RowCol object, 216–218
 RowColToPosition (method), 197
 RunMacro (method), 254

S

SaveScreen (method), 62
 SaveWorkSpaceAs (method), 46
 Scheme (property), 105
 ScreenTrigger object, 219–222
 ScreenTriggers (method), 255
 ScreenTriggers object, 223–227
 Security (property), 75
 Select (method), 63
 SelectAll (method), 64
 Send (method), 106
 SendString (method), 198
 SendStringTimeout (property), 201
 SenseCodes, 295
 Session object, 228–266
 Session.SetConnectParam Method, 294
 SessionDir (method), 47
 SessionInfo (property), 256
 SessionInfo object, 275–283
 SessionInfos (method), 47
 SessionInfos object, 284–287
 Sessions (method), 48
 Sessions object, 267–274, 294–??
 SetConnectParam (method), 257
 SetCursorLocation (method), 201
 SetData (method)
 Field object, 91
 PS object, 203
 SetPosition (method), 65
 SetProperties (method), 258
 SetSharing (method), 260
 SetSharing Command, 16
 SetSize (method), 66
 ShortName (property), 280
 ShowState (property), 67
 Size (property), 116
 SpaceAlloc (property), 107
 SpaceIncrement (property), 108
 SpaceQuantity (property), 109
 StartHostNotification (method), 204
 StartKeyIntercept (method), 205
 State (method), 179

State (property), 281
 Status (method)
 FileTransfer object, 110
 HostBrowser object, 127
 StopHostNotification (method), 207
 StopKeyIntercept (method), 207
 SubstituteChar (property), 208
 SystemAvailable (property), 173

T

TerminalType, 294
 Timeout (property)
 FileTransfer object, 111
 HostBrowser object, 128
 TimeStamp (property), 149
 TimingMarkRspTimeout, 295
 Tn3270Only, 295
 TraceFile (method), 49
 Translate (property), 112
 Turquoise (property), 144
 Type (property), 282
 Type libraries, 13-??

U

UnblockClose (method), 263
 UnblockPlay (method), 264
 UnblockUserInput (method), 265
 User-defined types, 35
 UserOptions (property), 113

Using ActiveX automation to open or close Aviva sessions, 15
 Using Aviva's type libraries, 13
 Using the SetSharing command to control an Aviva session, 16

V

Visible (property), 68
 VT/SSH Display Parameters, 312
 VT/Telnet Display Parameters, 309

W

WaitCursorAt (method), 209
 WaitCursorMove (method), 211
 WaitForString (method), 213
 WaitHostSettle (method), 214
 White (property), 144
 Width (property), 117
 WorkspaceDir (method), 50

X

X (property), 69

Y

Y (property), 70
 Yellow (property), 145