



**Q PlexView Edit**



## Table of Contents

QPlexView Overview .....	1
A Simple Example .....	3
How QPlexView Works .....	3
The ASP Script .....	5
Global.asa .....	9
QPlexView Host Connection .....	9
Properties and Methods .....	11
The QPlexView ActiveX Component Reference.....	11
Properties.....	11
Where to Look for the Trace Files: .....	15
Methods .....	15
AddPCXlate1, AddPCXlate2 and AddPCXlate3 .....	16
CloseSession.....	16
DoDataKey .....	16
DoTerminalKey .....	16
GetScreenAttribute .....	16
GetScreenColor .....	17
GetScreenLine .....	17
GetScreenText .....	17
OpenSession.....	18
OpenSessionStation.....	18
SetCursorPosition .....	19
SetScreenText .....	19
Wait .....	19
WaitForString.....	19
WaitForStringNot.....	20
Key Code Values .....	20
UTS Keys: .....	20
T27 Keys:.....	22
The Script Editor Dialogs.....	25
QPlexView Script Prototype Editor/Tester dialog .....	25
File menu .....	25
Edit menu.....	26
Search menu .....	26
Bookmarks .....	27
Options menu .....	27
Window.....	27
Help .....	28
Editor Properties .....	28
Edit Window Font .....	28
Tab Size.....	28
Highlight Colors.....	28
OK.....	29
Cancel .....	29
Help .....	29

The Dialog Designer .....	31
eXpress Plus Dialog Designer .....	31
File Menu .....	31
Edit Menu .....	31
Window Menu .....	32
Help .....	32
Toolbar Buttons.....	33
Grid Options .....	33
Show Grid .....	33
Snap to Grid .....	33
Grid Size .....	34
Alignment .....	34
Horizontal.....	34
Vertical .....	34
Size.....	34
Width.....	34
Height.....	34
Index .....	35

**QPlexView Overview**

QPlexView is a companion product of QPlex Client, the Web-based connectivity software from KMSYS Worldwide, Inc. QPlexView is an ActiveX component that provides a mechanism to Web-enable legacy applications running on Unisys mainframes. QPlexView is available in two versions: one for UTS terminal connections to 2200 family hosts and one for T27 connectivity to A-Series hosts. All references to QPlexView used here imply both versions.

QPlexView is designed for use in Microsoft's Active Server Page (ASP) environment. ASP is a server-side scripting environment for developing dynamic, interactive Web applications. ASP Scripts are written using a combination of HTML, scripting languages and COM components. ASP can use several scripting languages, but Microsoft Visual Basic Scripting Edition (VBScript) and Microsoft JScript are both included with ASP.

For more information on ASP, see Introduction to Active Server Pages from Anchor Productions, Inc.

Also, go to [msdn.microsoft.com](http://msdn.microsoft.com) and use the MSDN search engine to look for "Active Server Pages".

QPlexView ActiveX components are created within an ASP using VBScript or JScript whenever access to a legacy mainframe application is needed.

QPlexView provides a terminal-level interface to the legacy application's normal screens, thus no change to the legacy application is required.

Unlike QPlex Client, which provides terminal emulation via a Web page, QPlexView runs on the Web Server. Only the HTML generated by the ASP is sent to the client PC (Web Browser).

QPlexView can also be run on the client side of an application. It is kind of like QPlex Client with a read-only screen. All terminal interaction has to be done programmatically in whatever language is hosting the QPlexView OCX.

In addition to running in a client side application, QPlexView also supports all the printing capabilities of QPlex Client. The QPlex Client configuration program can be used to generate printer settings for use in QPlexView. See the ClientMode and Printer properties below. Also, see the AddPCXlate, RefreshScreen and Send methods below.



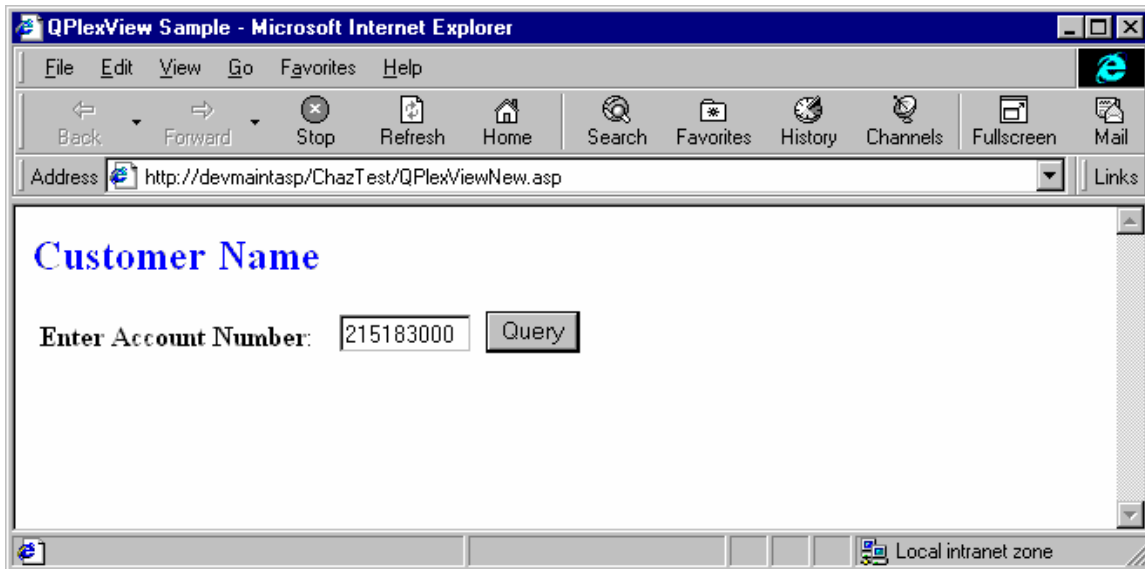
## A Simple Example

### How QPlexView Works

The best way to explain how QPlexView works is by example. The example shown here is an ASP where a legacy, host application screen is accessed to retrieve a customer's name when an account number is entered by the end user.

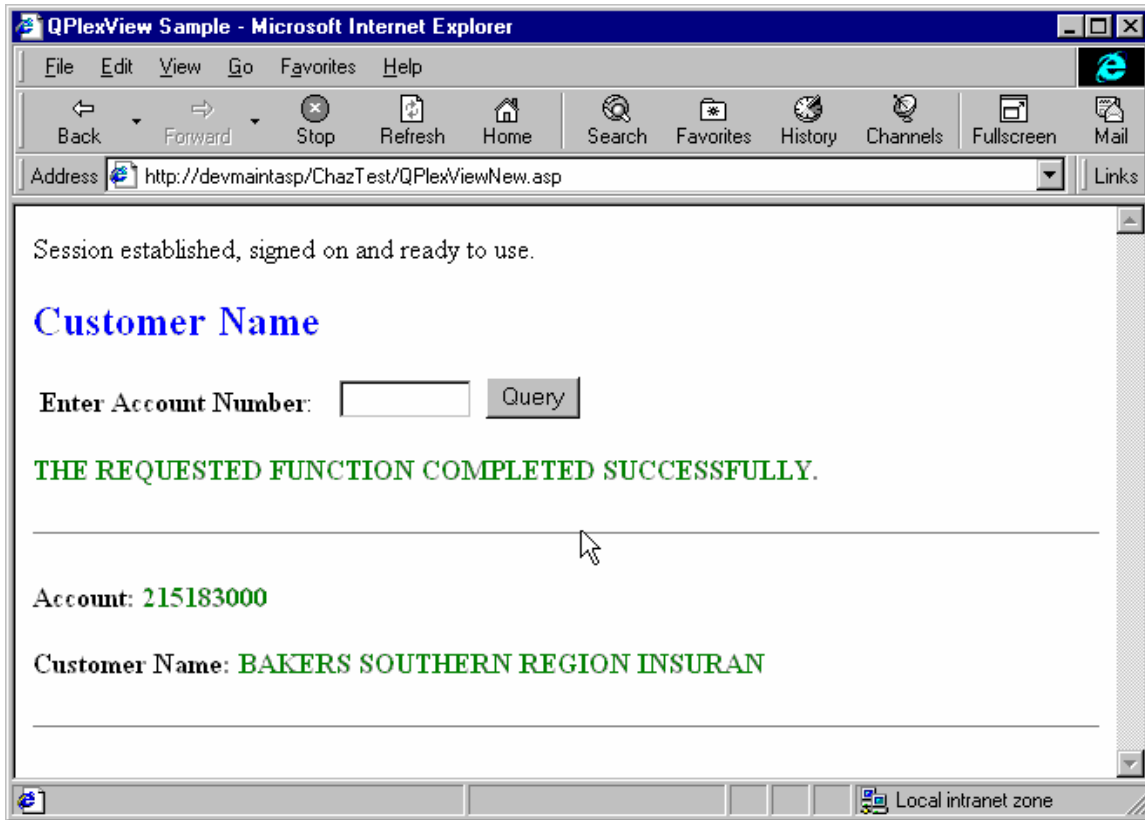
The end user only sees the generated Web page in his browser. All the underlying ASP script and host access is invisible because it runs completely on the Web sever — not the client machine.

The initial Web page seen by the end user appears as follows. Notice that this browser page is the only thing seen by and accessible to the end user.



**Figure 1: Initial (blank) Page**

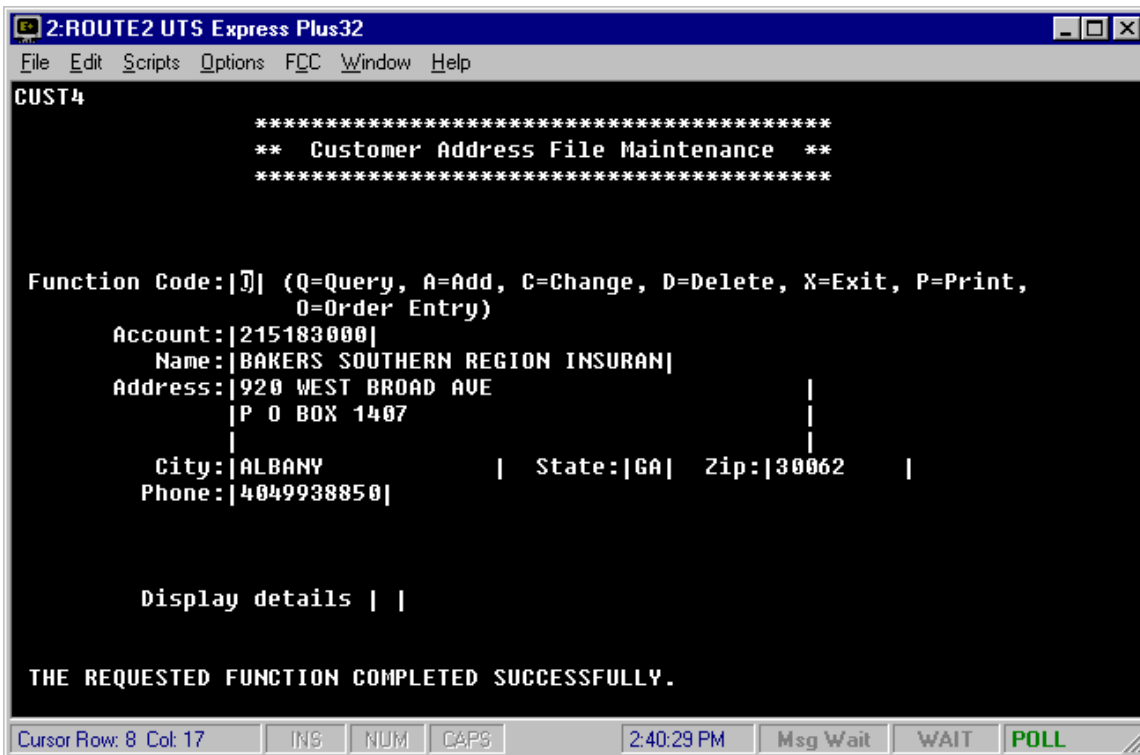
Upon successful completion of a Query, the Web page may appear as follows:



**Figure 2: Web Page after Successful Query**

The QPlexView ActiveX component actually sees the legacy application as if the user was running it from a dumb terminal. Properties and methods of the QPlexView component provide the mechanism that allows the ASP to interact with the legacy application screen.





**Figure 3: The Legacy Transaction Screen**

In this particular screen there is much more information available, but only the customer name is being retrieved to keep the example as simple as possible.

**The ASP Script**

The following is a short primer on web application development. Web page access is very similar to running a TIP transaction. Each access is independent of the previous and the next access. ASP scripts introduce the concept of a session. Information can be stored into session variables and retrieved later in another page’s script much like the XXX area in DPS can be used to store information between transactions.

Any information that is not saved at the session level must be re-created by every page access. In the case of ASP controls, such as QPlexView, creating and destroying the control at the beginning and ending of each page can be very resource intensive.

The code at the beginning of the sample ASP script makes use of session variables to allow re-use of the QPlexView control.

As mentioned earlier, the ASP script combines both HTML and VBScript or JScript. In this case, VBScript was used. All VBScript portions of the ASP are enclosed between the <% and %> HTML sequences. The amount of VBScript enclosed can be anything from a simple VB variable name to several lines of VBScript code. The following ASP was developed using Microsoft Front Page:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>QPlexView Sample</title>
```

```

</head>
<%
'
Response.Buffer = True

if Request("Btn_Query") = "Query" then
'
' Look for and or construct QPlexView Object and establish session
'
'
If IsObject(Session.Contents("SQPlexView")) then
' The line above determines if a QPlexView control has been created.
' Session.Contents is a container of all variables and objects stored at the
' session level. IsObject is a method that determines if its argument is an
' object (instantuated control) or not. Since session variables are local to a
' single session, if SQPlexView is an object we should re-use it, because it is
' a copy of the QPlexView control that we have already instantuated..
' Response.Write "Existing QPlexView object found<BR>"
' The line above, if not commented out, would display a line in the resulting
' output page, that would indicate we found an existing control. This is the
' equivalent to using a COBOL DISPLAY statement for debugging.
If Session("SQPlexView").SessionOpen = 1 Then
' Since we are re-using a control, we should see if we already have a session
' open to the 2200. (In actual usage, we would most likely have the session open,
' but this is just more code to make the application more reliable.
' Response.Write "Existing session open<BR>"
' Once again a debug display.
OpenStatus = 1
' Indicate that we have a valid session to the 2200.
Else
OpenStatus = Session("SQPlexView").OpenSession("KMSMCB",
Request.ServerVariables("SERVER_NAME"), 102)
' We did not have a session so the line above is used to open a session. The
' code 'Request.ServerVariables("SERVER_NAME")' is used to retrieve the name of
' the web server that is running the script on this page. This makes the ASP
' code portable, if the Host Gateway Server that we are connecting through is on the
' same machine.
Session("SignedOn") = "N"
' Indicate that we have not signed-on to the 2200. This would only be necessary
' if you are running DEMAND or TIP with TIP Session Control enabled.
End If
Else
' Response.Write "Creating QPlexView object<BR>"
' Once again a debug display.
Set Session("SQPlexView") = Server.CreateObject("QPlexViewUTSx.QPlexViewUTS")
OpenStatus = Session("SQPlexView").OpenSession("KMSMCB",
Request.ServerVariables("SERVER_NAME"), 102)
' The line above is used to open a session to the 2200. See earlier occurrence
' for more information.
Session("SignedOn") = "N"
End if
if OpenStatus = 1 then
' The line above is used to check if we have a session to the 2200. If not
' then we know that the OpenSession call above failed and we should display
' an error message. The OpenStatus is not a session variable because we can
' determine the session status from the control by checking the SessionOpen
' property.
' Response.Write "Session opened<BR>"
' Once again a debug display.
If Session("SignedOn") <> "Y" Then
Rslt = Session("SQPlexView").WaitForString(2,23,"Enter your user-id/password",5000)
' The line above will wait for the userid and password prompt.
If Rslt = 0 Then
Msg = "Didn't get 'Enter your user-id/password' string<BR>"
Else
Rslt = Session("SQPlexView").WaitForString(2,24," ",5000)
Session("SQPlexView").DoTerminalKey 23 ' CURSOR_TO_HOME
Session("SQPlexView").DoTerminalKey 14 ' ERASE_DISPLAY
Session("SQPlexView").SetScreenText 1, 1, "QPLEX/DEMO"
Session("SQPlexView").DoTerminalKey 66 ' CURSOR_TO_END_LINE
Session("SQPlexView").DoTerminalKey 36 ' TRANSMIT_KEY

```

```

Rslt = Session("SQPlexView").WaitForString(2,23,"Previous session was:",5000)
If Rslt = 0 Then
  Msg = "Didn't get signed on<BR>"
Else
  Session("SignedOn") = "Y"
  ' We have a session open to the 2200 and, if required,
  ' signed on. Now we can run our first transaction. The
  ' code here should be used to create input to run
  ' the transaction, look at the output and generate
  ' the HTML for information display.
End If
End If
Else
  Msg = "Previous session found to be signed on.<BR>"
Response.Write Msg
  ' The line above will display the contents of the message variable. This is used
  ' as the primary means of indicating to the user the state of connection and
  ' data transfer.
End If
If Session("SignedOn") = "Y" Then
  Response.Write "Session established, signed on and ready to use.<BR>"
%>
<!--Comment. Put HTML code here and more script code here to get data
from the control-->
<%
  ' Run initial transaction to get blank screen form
  Session("SQPlexView").DoTerminalKey 23 ' CURSOR_TO_HOME
  Session("SQPlexView").DoTerminalKey 14 ' ERASE_DISPLAY
  Session("SQPlexView").SetScreenText 1, 1, "CUST3"
  Session("SQPlexView").DoTerminalKey 66 ' CURSOR_TO_END_LINE
  Session("SQPlexView").DoTerminalKey 36 ' TRANSMIT_KEY
  Rslt = Session("SQPlexView").WaitForString(1, 1, "CUST4 ", 5000)
  If Rslt = 0 Then
    Msg = "Didn't get CUST4 Screen"
  Else
    ' Enter the Query function code anf account number and transmit
    Session("SQPlexView").SetScreenText 17, 8, "Q"
    Session("SQPlexView").SetScreenText 17, 10, Request("TextAccount")
    Session("SQPlexView").SetScreenText 1, 23, " "
    Session("SQPlexView").SetCursorPosition 17, 11
    Session("SQPlexView").DoTerminalKey 36 ' TRANSMIT_KEY
    rslt = Session("SQPlexView").WaitForStringNot(1, 23, " ", 5000)
    If Rslt = 0 Then
      Msg = "No response to query"
    Else
      ' Get response from the screen.
      Msg = Session("SQPlexView").GetScreenText(2, 23, 78)
      ' Status message at bottom of screen
      Account = Session("SQPlexView").GetScreenText(17, 10, 9) ' Account number
      CustName = Session("SQPlexView").GetScreenText(17, 11, 30) ' Customer name
      QueryDone = True
    End If
  End If
End If

Else
  Response.Write "Could not establish a session with the 2200 host.<BR>"
End If

  ' At this point the script has completed. This page can be accessed again with
  ' different input values to force it to do a query on specific data etc. It will
  ' re-use the same control and open session to the 2200 for each use. You must
  ' have a method for terminating the session to the 2200 and destroying the
  ' QPlexView control within the ASP script. This action could be accomplished by
  ' querying the value of a button that is used to activate the page and script.
  ' The action would be accomplished by the following code.
  Session("SQPlexView").CloseSession()
  ' There is not way for the browser to indicate to ASP that it's session is to be
  ' terminated. Sessions will terminate when their idle time expires. (This value
  ' is set in the Application configuration in IIS.) When the session terminates,

```

```

'      code in Global.asa will be called (see below).
End If
BtnVal = "Query"
%>

<body>

<h2><font color="#0000FF">Customer Name</font></h2>

<form method="POST" action="QPlexViewNew.asp"
onsubmit="return FrontPage_Form1_Validator(this)" name="FrontPage_Form1">
  <input type="hidden" name="VTI-GROUP" value="0"><p>&nbsp;<strong>Enter Account
Number</strong>:&nbsp;  <input type="text" name="TextAccount" size="9" maxlength="9">&nbsp;  <input type="submit"
value="<%=BtnVal%>" name="Btn_Query"></p>
</form>

<h4><font color="#008000"><%= Msg%></font></h4>
<%if QueryDone = True then%>

<hr>

<h4>Account:<font color="#008000"> <%=Account%></font></h4>

<h4>Customer Name:<font color="#008000"> <%=CustName%></font></h4>

<hr>
<%End If%>
</body>
</html>

```

Note the <% sequence that starts several lines of VBScript code.

The first "if" statement (*if Request("Btn\_Query") = "Query" then*) acts as a first time switch, if the caption of Btn\_Query is equal to "Query", the rest of the script will be execute. On first entry, the caption of Btn\_Query will be blank so the rest of the VBScript will not be executed and only the initial HTML will display a blank input form to the end user (Figure 1).

The rest of the VBScript at the beginning of the ASP contains all the logic required to complete the transaction. In this case, a session is opened to a Unisys 2200 host and a user-id and password is entered to establish a TIP session. Next, a transaction code (CUST3) is entered into the screen and transmitted to the host to get a blank screen for the transaction required for this function. Once the blank screen is detected, the Query function code (Q) and the user-supplied account number are entered into the screen in the appropriate fields and transmitted to the host. Upon receiving the response, the desired information is placed into VBScript variables (Msg, Account and CustName) that are used later in the HTML resulting in a completed Web page (Figure 2).

This is a very simple example, but QPlexView can be used to perform very complex transaction sequences involving the navigation of many screens and even performing updates. This example and a second more complex ASP are installed with QplexView and normally may be found in the c:\Program Files\KMSystems\QplexView\Samples installation directory.

Remember that the host is only accessed by the QPlexView ActiveX component running on the Web server. The end user never has any knowledge of how the information was obtained. QPlexView and ASP provide a very secure method of providing legacy application access to users across the World Wide Web.

**Global.asa**

Code in Global.asa gets control 4 different times.

- 1) – When an Application is initialized.
- 2) – When an Application is terminated.
- 3) – When a Session is initialized.
- 4) – When a Session is terminated.

```
<script LANGUAGE="VBScript" RUNAT="Server">
SUB Application_OnStart
  Application("ApplicationName") = "QPlexView"
  ` Line above sets an Application level variable (similar to a Session level
  ` variable) so that the name of the Application can be determined. This is
  ` mostly for debug purposes.
END SUB
</script>

<script LANGUAGE="VBScript"
RUNAT="Server">
SUB Application_OnEnd
END SUB
</script>

<script LANGUAGE="VBScript" RUNAT="Server">
SUB Session_OnStart
  ` This routine is called when a new session is started. It is a good place to initialize
  ` variables that all users of an application will need.
END SUB
</script>

<script
LANGUAGE="VBScript" RUNAT="Server">
SUB Session_OnEnd
  ` This routine is called when a new session is terminated. Sessions can be terminated under
  ` two conditions.
  ` 1) - The session idle time is exceeded.
  ` 2) - The ASP script requests that the session be aborted.
  ` We have clean-up code here, because there is no way to
  ` know the state of the QPlexView control at time of termination.
  if IsObject(Session.Contents("SQPlexView")) then
  ` Determine if we have a control.
    if Session("SQPlexView").SessionOpen = 1 then
  ` Determine if we have a session to the 2200. (We don't care if we are
  ` signed-on to the 2200 because the session close will terminate the
  ` sign-on.
      Session("SQPlexView").CloseSession()
  ` Close the session.
    end if
    Set Session("SQPlexView") = Nothing
  ` This line is VERY important. This is the only way to make the copy of the object that we
  ` just finished actually go away. If this is not done, memory will be orphaned until such
  ` time as IIS is stopped and re-started. Do not put this line of code into another web page.
  ` Doing so will cause numerous problems.
  end if
END SUB
</script>
```

**QPlexView Host Connection**

QPlexView connects to the host through the Host Gateway Server in the same manner as QPlex Client.



## Properties and Methods

### The QPlexView ActiveX Component Reference

This section defines the properties and methods provided by the QPlexView ActiveX components.

#### Properties

Property	Type	Description
AllowUserFKey	Integer Read/Write	Default is False (0). Controls the end users ability to do function keys (a form of transmit). If set to 0 (False), the script must perform this function. If set to 1 (True), the user can initiate this function; however, there is no way for the script to detect when or if the function has occurred.  This property may be set when running in ClientMode and should be set before the OpenSession method is called.
AllowUserTransmit	Integer Read/Write	Default is False (0). Controls the end users ability to do a transmit. If set to 0 (False), the script must perform this function. If set to 1 (True), the user can initiate this function; however, there is no way for the script to detect when or if the function has occurred.  This property may be set when running in ClientMode and should be set before the OpenSession method is called.
AutoScaleMode	Integer Read/Write	0 = No autoscale, 1 = Size screen to font size, 2 = Size font to fill screen (default).  This property may be set when running in ClientMode and should be set before the OpenSession method is called.
BasicLogging	Integer Write Only	Default is 0. Set this property to 1 to get basic Windows/NT logging of actions performed by the QPlexView component.
ClientMode	Integer Read/Write	A value of 1 indicates that QPlexView is running on the client side and is allowed to print. Defaults to 0 indicating server mode operation.
ColorSettings	String Read/Write	A string that specifies the configured color settings.  This property may be set when running in ClientMode and should be set before the OpenSession method is called.
CursorColumn	Integer Read Only	The current cursor column. If a session is not open, a 0 is returned.

<b>Property</b>	<b>Type</b>	<b>Description</b>
CursorRow	Integer Read Only	The current cursor row. If a session is not open, a 0 is returned.
DatacomOptions (T27 only)	String Read/Write	This property contains a string of settings that are generated in the QPlex Client T27 User Configuration Manager (QPlexCfgT27.exe) and govern communications between the host and PC. The WEB developer can generate the HTML for then settings the copy/paste them into the ASP script.
DetailLogging	Integer Write Only	Default is 0. Set this property to 1 to get detailed Windows/NT logging of actions performed by the QPlexView component.
HGSSecurityFlags	Integer Read/Write	0 = None 1 = Authenticate 3 = Authenticate and encrypted
HostGatewayType	Integer Read/Write	Set to "1" for HGS, and "0" (default) for QPlex. See the QuickStartHGS.asp for an example.
KeyboardLocked (UTS only)	Integer Read only	Indicates if the keyboard is locked. 1 = Keyboard locked, 0 = unlocked.
KeyboardOptions (T27 only)	String Read/Write	This property contains a string of settings that are generated in the QPlex Client T27 User Configuration Manager (QPlexCfgT27.exe) and govern keyboard actions. The WEB developer can generate the HTML for the settings then copy/paste them into the ASP script.
LastErrorCode	Integer Read Only	This property can be displayed in the HTML after an error is encountered. Do not use this property to determine whether an error has occurred, as the property is not cleared after the error.
LastErrorMessage	String Read Only	This property can be displayed in the HTML after an error is encountered. Do not use this property to determine whether an error has occurred, as the property is not cleared after the error.
MessageWaiting (UTS only)	Integer Read Only	Indicates if message waiting is set. 1 = message waiting set, 0 = not set.
Pages (UTS only)	Integer Read/Write	This property may be set to enable the paging feature of QPlexView.  You may specify from one (default) to nine screen pages for each UTS screen. In conjunction with the paging feature, the PAGE DOWN and PAGE UP UTS keys are mapped to the Page Down and Page Up keyboard keys, respectively. Note: You must first click



Property	Type	Description
		<p>somewhere on the screen page before utilizing the Page Up/Page Down keys.</p> <p>Paging is most useful where only one screen is available. Paging provides the means to maintain multiple output screen pages for reference while running additional transactions.</p> <p><b>WARNING:</b> The host is totally unaware of the paging feature. Applications, especially DPS, often check the screen input for specific context, meaning the application thinks a certain screen was displayed and expects data to come from that screen. Use of the Page Down and Page Up keys with this feature can leave the screen in a state not expected by the application.</p>
Printer1 Printer2 Printer3 (UTS only)	String Read/Write	<p>A string that specifies the configured printer settings. There are 3 different printer properties and methods (see the AddPCXlate methods, below) for each corresponding Device Id. (DID) for printer selection by the host. The DIDs are always assigned as follows: 1 = x73, 2 = x74 and 3 = x75.</p> <p>All printer settings are the same for all QPlex Session screens. That is, you cannot configure different printers for each screen.</p> <p>The String values used in the PrinterX property methods are generated using the configuration program.</p>
PrintTimeOut	Integer Read/Write	<p>-1 (or any negative value) = No timeout, 0 = Timeout immediately after printing, <i>n</i> (any positive number) = Timeout <i>n</i> seconds after printing.</p> <p>The default is 15 seconds.</p>
ReadOnly	Integer Read/Write	<p>Default is 1 (True). ReadOnly controls whether or not the end-user is allowed to type directly into the screen. If false the only way to enter anything in the screen is programmatically in the script.</p>
ScreenFont	String Read/Write	<p>A string that specifies the configured screen font, style and size.</p> <p>This property may be set when running in ClientMode and should be set before the OpenSession method is called.</p>
SessionOpen	Integer Read Only	<p>Indicates whether a host session is currently open. 1 = Open, 0 = Not Open.</p>
ShowInactivePage	Integer	<p>Use to show the inactive page screen. 1 = show,</p>

Property	Type	Description
	Read/Write	<p>0 = do not show.</p> <p>Use this control to toggle the inactive screen page display. When checked along with the screen paging feature enabled, the last screen page will be shown in an inactive screen page window located in the lower half of the screen window. This feature allows you to view the results of a transaction or program in an inactive, read-only, pane of the window while executing another transaction in the active portion of the window. The inactive pane is always shown in black on white.</p> <p>If there is more than one active screen page configured, clicking the gray separator bar causes the inactive screen page display to scroll to the next inactive screen page (notice that the current active page is never shown in the inactive pane).</p> <p>When using this option, the screen window may have to be resized manually to accommodate the dual screen display.</p> <p>Note: The inactive page will not appear until the user does the first Page Down or Page Up.</p> <p><b>WARNING:</b> The host is totally unaware of the paging feature. Applications, especially DPS, often check the screen input for specific context, meaning the application thinks a certain screen was displayed and expects data to come from that screen. Use of the Page Down and Page Up keys with this feature can leave the screen in a state not expected by the application.</p>
ShowStatusBar	Integer Write only	Use to show a status bar at the bottom of the screen. 1 = show, 0 = do not show.
SITAOpenId (UTS only)	Integer Read/Write	Indicates if the Special SITA Gabriel Handling option is to be used. Values can be 0 (default) or 1. 1 means use the SITA communications options.
TraceOption	Integer Write Only	<p>Default is 0 (none). Set this property to 2 to a trace to file. Registry entries must be manually created on the machine where QPlexView is running (the web server) as follows:</p> <p>Please see "Where to Look for the Trace Files" below.</p>
TransmitType	Integer Write Only	Set this property to the transmit type expected by the host software. Allowed values are VAR, CHAN, ALL. Most host applications expect VAR.
TransportTrace	Integer	Default is 0 (none). To start a Transport trace

Property	Type	Description
	Write Only	the new property TransportTrace is set to 1. This property can be set at any point, but it is recommended that it be set before the OpenSession. Please see "Where to Look for the Trace Files" below.
UTSSettings (UTS only)	String Read/Write	A string that specifies the configured UTS settings. This property may be set when running in ClientMode and should be set before the OpenSession method is called.
Version	String Read Only	Use to retrieve the current version of the QPlexView ActiveX component.
VideoOptions (T27 only)	String Read/Write	This property contains a string of settings that are generated in the QPlex Client T27 User Configuration Manager (QPlexCfgT27.exe) and govern the state of the environment display. The WEB developer can generate the HTML for the settings then copy/paste them into the ASP script.

### Where to Look for the Trace Files:

The software uses a Windows registry key for a default eXpress trace directory. The key is "Software\KMSystems\eXpressTrace" under HKEY\_LOCAL\_MACHINE. The value name is "DefaultTraceFilePath". The value data is initially null and must be MANUALLY entered as a standard drive and path specification (e.g., "C:\trace").

Note: The account that the web service is running under must have read access to the registry key and modify access to the directory specified in the registry key.

If no default directory is specified and the application is not running as a service (QPlexView runs as a service), then a standard File Save dialog will be displayed.

If the user cancels the File Save dialog or the trace file cannot be created in the specified directory, the trace will not be performed. A message will be displayed if not running as a service.

In the case of a server application (QPlexView) where no dialogs or messages can be used, the default directory must be specified in the registry and default file names will be used. The default file names are: "KMSTransExnnnnnnnnn.trc" for a transport trace, "eXpressTracennnnnnnnn.txt" for an emulator trace and "ETEMTracennnnnnnnn.txt" for a T27 ETEM trace, where nnnnnnnnn represents a unique date/time stamp.

## Methods

### AddKeyDef

**Procedure** AddKeyDef (*KeyDefinitionString*)

*Description:*

This procedure allows the keyboard to be customized from the Web page. The *KeyDefinitionString* is generated by the QPlex Client Configuration Manager and pasted into the script (see the QPlex Client Configuration Manager help in the QPlex Client).

**AddPCXlate1, AddPCXlate2 and AddPCXlate3**

Sub **AddPCXlate1** (*PrinterSettings* as String)

Sub **AddPCXlate2** (*PrinterSettings* as String)

Sub **AddPCXlate3** (*PrinterSettings* as String)

*Description:*

These procedures allows printer character translation from the Web page. There are 3 different printer methods for each corresponding Device Id. (DID) for printer selection by the host. The DIDs are always assigned as follows: 1 = x73, 2 = x74 and 3 = x75.

The String values used in the AddPCXlate methods are generated using the Configuration Manager.

**CloseSession**

Function **CloseSession** () as Integer

*Description:*

Close the current host session. If successful, this method returns a 1; else, it returns a 0

**DoDataKey**

Sub **DoDataKey** (*KeyCode* as Integer)

*Description:*

Enter a data key into the internal terminal emulator as if it were typed from the keyboard by an end user. *KeyCode* is the ASCII character value (ex A = 65).

**DoTerminalKey**

Sub **DoTerminalKey** (*KeyCode* as integer)

*Description:*

Cause the specified Terminal Key sequence to be executed by the internal terminal emulator. All terminal key functions are available (see Key Code Values).

Note: Use the DoTerminalKey method to do a transmit, send a function key, erase display, etc.

**GetScreenAttribute**

Function GetScreenAttributes(*Column* as Integer, *Row* as Integer) as Integer

*Description:*

Retrieve certain screen attributes at the specified screen row and column coordinates. The following attribute values may be returned:

<b>Constant</b>	<b>Value</b>	<b>Description</b>
SATTR_NORMAL	0	Normal field

<b>Constant</b>	<b>Value</b>	<b>Description</b>
SATTR_FIELD	1	Start of field (set on 1st position of field)
SATTR_TAB	2	Tab stop (at start of field only)
SATTR_PROTECTED	8	Protected-Output only
SATTR_VIDEO_OFF	16	Video off (Data is present in screen buffer)
SATTR_BLINK	128	Blinking field
SATTR_RIGHT	256	Right justified data
SATTR_REV	1024	Reverse video

*Unique T27**Attributes:*

SATTR_BRIGHT	512	T27 Bright
SATTR_ULINE	4	T27 Underline

*Unique UTS**Attributes:*

SATTR_NUMERIC	32	Numeric only input
SATTR_ALPHA	64	Alpha only input
SATTR_LOWINT	512	UTS Low intensity
SATTR_CHANGED	4	Data field changed flag

*Returns:*

The function returns the attribute bits. If an error is encountered, a 0 is returned.

**GetScreenColor**

Function **GetScreenColor**(Column as Integer, Row as Integer) as Integer

*Description:*

Retrieve the color attributes of the screen at the specified screen row and column. The following color attribute codes are used:

*Returns:*

This function returns the color attribute code. If an error was encountered, a 0 is returned.

**GetScreenLine**

Function **GetScreenLine** (Row as Integer) as String

*Description:*

Retrieve an entire line of text, including leading and trailing spaces, from the internal screen at the specified *row*. If successful, the function returns the line as a string; otherwise, it returns an empty string.

**GetScreenText**

Function **GetScreenText** (Column as Integer, Row as Integer, Length as Integer) as String

*Description:*

Retrieve a string of text, including leading and trailing spaces, from the internal screen at the specified *row* and *column* and for the specified *length*. If successful, the function returns the text as a string; otherwise, it returns an empty string.

### **HoldMessages**

Sub HoldMessages

#### *Description:*

Force messages from the host to be held until a Receive, UnholdMessages is issued. Note: This subroutine only applies to UTS emulation.

### **OpenSession**

Function **OpenSession** (*OpenId* as String, *IPAddress* as String, *IPPort* as Integer) as Integer

#### *Description:*

Open a host session using the specified *OpenId*, Host Gateway Server *IPAddress* and Host Gateway Server *IPPort*. This method returns a 1 if successful; else, it returns a 0.

### **OpenSessionStation**

Function **OpenSessionStation** (*OpenId* as String, *IPAddress* as String, *IPPort* as Integer, *StationName* as String) as Integer

#### *Description:*

This method is used in lieu of the OpenStation method when you want to specify the specific station name to use for the open.

Open a host session using the specified *OpenId*, Host Gateway Server *IPAddress*, Host Gateway Server *IPPort* and *StationName*. This method returns a 1 if successful; else, it returns a 0.

### **Receive**

Function **Receive**(*TimeoutValue* as Integer) as Integer

#### *Description:*

The Receive method informs the user that a message, containing text and/or control sequences, has been received from the transport and mapped to the internal screen. The Receive method does **NOT** cause anything to actually happen, because the UTS terminal (which is being emulated) receives data without any action taken by the user. To retrieve the actual text received, use the GetScreenText or GetScreenLine method.

When the Receive function is called and used in conjunction with the HoldMessages method, one message will be accepted and processed (moved to the screen, etc.).

Used with the HosdMessages method, this function is useful when a stream of messages are expected from the host and the script needs to process each one separately. Receiving each message explicitly guarantees that no messages will be missed due to the asynchronous nature of message receipt from the host.

The *TimeoutValue* specifies, in milliseconds, how long the Receive should wait for a message to arrive before returning.

#### *Returns:*

If no messages are received from the host within the specified `TimeOut` value, the function will return a `False` (zero). If a message is received the function returns as `True` (-1).

### **RefreshScreen**

Sub `RefreshScreen`

*Description:*

Forces the visible screen to be repainted. Should be called after entering text into the screen or moving the cursor. Only use in `ClientMode`.

### **Send**

Function `Send(TextToSend as Sstring) as Integer`

*Description:*

The `Send` method transmits a specified string directly to the transport without affecting the content of the screen. The string is passed unmodified—no control sequences are added. Keep in mind that the host application may not function correctly using this method, because it may require control sequences that separate fields, messages, etc. Normally, when data is sent from the screen to the host, it will contain additional control sequences (FCCs, field separation, start-of-entry position, etc.) which may be essential to the host application.

*Returns:*

The return value can be 0 or 1. A return of 0 means the send was not done because the session is not currently open. A return of 1 indicates the `Send` completed at the transport level.

### **SetCursorPosition**

Sub `SetCursorPosition (Column as Integer, Row as Integer)`

*Description:*

Move the screen cursor to the specified *row* and *column* in the internal screen.

### **SetScreenText**

Sub `SetScreenText (Column as Integer, Row as Integer, Text as String)`

*Description:*

Replace *text* in the internal screen starting at the specified *row* and *column*.

### **UnholdMessages**

Sub `UnholdMessages`

*Description:*

Releases messages stopped by the `HoldMessages` statement. Messages will be received and processed in the normal manner.

### **Wait**

Sub `Wait (WaitTime as Integer)`

*Description:*

`Wait` does a timed wait for the specified *WaitTime*. The *WaitTime* is specified in milliseconds.

### **WaitForString**

Function **WaitForString** (*Column* as Integer, *Row* as Integer, *Text* as String, *TimeOut* as Integer) as Integer

*Description:*

Wait for specified *text* to appear in the screen at the specified *column* and *row* position. If the *text* appears before the *TimeOut* expires, this function returns the value 1; otherwise, it returns a 0.

### WaitForStringNot

Function **WaitForStringNot** (*Column* as Integer, *Row* as Integer, *Text* as String, *TimeOut* as Integer) as Integer

*Description:*

Wait for specified *text* to be no longer present in the screen at the specified *column* and *row* position. If the *text* changes before the *TimeOut* expires, this function returns the value 1; otherwise, it returns a 0.

### Key Code Values

The following list of key code values may be used with on the DoTerminalKey method. Note that the same list may be displayed in a separate window when working with the QPlexView Script Editor (see View Terminal Key Code Values on the Options menu).

#### UTS Keys:

Code Value	Key
95	BACK_SPACE
6	CURSOR_DOWN
7	CURSOR_LEFT
32	CURSOR_RETURN_KEY
8	CURSOR_RIGHT
66	CURSOR_TO_END_LINE
23	CURSOR_TO_HOME
65	CURSOR_TO_START_LINE
9	CURSOR_UP
11	DELETE_IN_DISPLAY
12	DELETE_IN_LINE
10	DELETE_LINE
67	ERASE_CHAR
14	ERASE_DISPLAY
15	ERASE_TO_END_DISPLAY
16	ERASE_TO_END_FIELD
17	ERASE_TO_END_LINE
43	FKEY_1
44	FKEY_2
45	FKEY_3
46	FKEY_4



<b>Code Value</b>	<b>Key</b>
47	FKEY_5
48	FKEY_6
49	FKEY_7
50	FKEY_8
51	FKEY_9
52	FKEY_10
53	FKEY_11
54	FKEY_12
55	FKEY_13
56	FKEY_14
57	FKEY_15
58	FKEY_16
59	FKEY_17
60	FKEY_18
61	FKEY_19
62	FKEY_20
63	FKEY_21
64	FKEY_22
25	INSERT_IN_DISPLAY
26	INSERT_IN_LINE
24	INSERT_LINE
27	KEYBOARD_UNLOCK
28	LINE_DUP
29	MSG_WAIT
30	PRINT_KEY
69	PRINT_ENTIRE_SCREEN
3	SOE
33	TAB_BACK
34	TAB_FORWARD
35	TAB_SET
36	TRANSMIT_KEY

**T27 Keys:**

<b>Code Value</b>	<b>Key</b>
249	ARROWDN
247	ARROWLEFT
248	ARROWRIGHT
246	ARROWUP
8	BACKSPACE
196	BACKTAB
218	BOUND
13	CARRIAGERTN
16442	CLRALLVTAB
134	CLREOL
135	CLREOP
159	CLRFORMS
128	CLRHOME
16432	COPY
164	CTRL
16431	CUT
234	DBLZERO
132	DELCHAR
16425	DELCHARPAGE
133	DELLINE
174	HOME
130	INSCHAR
16424	INSCHARPAGE
131	INSLINE
168	LOCAL
165	LOCKCTRL
16415	LOGICALEOL
217	MARK
138	MOVELINEDOWN
139	MOVELINEUP
253	NEXTPAGE
16434	PASTE
252	PREVPAGE
157	PRINTALL
156	PRINTUNPROT
214	RECALL
170	RECEIVE
136	ROLLDN
137	ROLLUP

<b>Code Value</b>	<b>Key</b>
158	SETFORMS
166	SPECIFY
213	STORE
198	TAB
141	TOGGLEFORMS
16441	TOGGLETAB
172	TRANSMIT
16428	TRANSMITLINE
236	TRIPZERO
210	UPPERONLYON
211	UPPERONLYUOFF
16426	WRITEESC
3	WRITEETX
16427	WRITEGS



## The Script Editor Dialogs

### QPlexView Script Prototype Editor/Tester dialog

This window provides the means to develop and edit QPlexView Scripts. The script editor contains a Multiple Document Interface (MDI) that allows more than one script to be edited at a time.

Note: The QPlexView Script Editor may be used to prototype and test portions of the ASP that contain the methods and properties of the ActiveX Component. It may not be used to prototype the entire ASP script. You will need to cut and paste from the QPlexView Script Editor to the application tool used to create the ASP.

Following is a description of each QPlexView Script Editor command. All the commands may be performed by making a menu selection. Toolbar buttons, shortcut keys and right mouse click actions are available for frequently used commands. A right mouse click anywhere in the test area will produce a pop-up menu of frequently used commands.

The menu items below show an image of the toolbar button and the shortcut key combination (in parentheses) where applicable.

#### File menu

The **File** menu contains commands to maintain script files and setup printing.



#### **New (Ctrl+N)**

Use this command to create a new script file (.BAS).



#### **Open (Ctrl+O)**

Use this command to open an existing script file (.BAS).



#### **Save (Ctrl+S)**

Use this command to save the current script file (.BAS).



#### **Save As...**

Use this command to save the current script file (.BAS) to another file name.

#### **Close**

Close the currently selected script.

#### **Close All**

Close all open scripts.



#### **Print (Ctrl+P)**

Use this command to print the entire script.

#### **Printer Setup...**

Allow margins to be set and allow printers and printer fonts to be selected for printing.

#### **Clear Previous File List**

Remove all file names from the list of previously accessed files.

Use this command to edit the properties of the script editor: window font, highlight colors and tab stops.

**Exit**

Exit the Script Editor.

**Previous File List**

Select (open) from the list of previous accessed script files.

**Edit menu**

The **Edit** menu contains commands to manage selected text between the editor and the Windows clipboard.

**Undo (Ctrl+Z)**

Use this command to reverse the effects of the most recent change.

**Redo (Ctrl+Shift+Z)**

Use this command to reverse the effects of the most recent **Undo** command.

**Cut (Ctrl+X)**

Use this command to place the selected text on the clipboard and delete.

**Copy (Ctrl+C)**

Use this command to copy the selected text to the clipboard.

**Paste (Ctrl+V)**

Use this command to paste the contents of the clipboard to the current cursor position.

**Delete (Ctrl+D)**

Use this command to delete the selected text without copying to the clipboard.

**Word Wrap (Ctrl+W)**

Use this command as a toggle. By default, long lines may only viewed/edited by first bringing the excess text into view with the horizontal scroll bar or by using the cursor keys (arrows). When **Word Wrap** is set, long lines wrap to the next line and are viewable within the confines (width) of the window.

**Syntax Check (F4)**

Use this command to check the syntax of the entire script.

**Run Script (F5)**

Use this command to run the script.

**Dialog Designer (F10)**

Use this command to initiate the Enable Dialog Designer. To edit an existing dialog, place the dialog on the Windows Clipboard before using this command.

**Search menu**

The **Search** menu contains commands to locate and change text within the script.

**Find... (Ctrl+F)**

Use this command to enable the **Find** dialog used to locate text strings.

**Find Again (F3)**

Use this command to find the next occurrence of the same string used on the previous find.

**Replace... (Ctrl+R)**

Use this command to enable the **Replace** dialog used to locate and replace text strings.

**Go to Line (Ctrl+G)**

Use this command to go to a specific line.

**Bookmarks**

The **Bookmarks** menu contains commands to mark lines and navigate within the script.

**Set Bookmark 1 through 5 (Shift+F1 through Shift+F5)**

Use one of these commands to mark a line at the current, text cursor position. A bookmarked line will appear with a gray background.

**Go to Bookmark 1 through 5 (Ctrl+F1 through Ctrl+F5)**

Use one of these commands to go to a line previously bookmarked by one of the five corresponding **Set Bookmark** commands.

**Options menu**

The **Option** menu contains commands to specify color, font and tab stop preferences.

**Show Tool Bar**

Use this command to toggle the display of the toolbar.

**Syntax Highlight**

Use this command to toggle the display of the script syntax. This command is affected by the settings of the **Editor Colors** command, below.

**View Terminal Key Code Values**

Use this command to show the acceptable key code values to be used with the DoTerminalKey subroutine. You browse the terminal key code values in read-only mode; however, you may copy code from the browser window to the Windows clipboard using the **Ctrl+C** key, and subsequently paste them into the script with the **Ctrl+V** key.

**Window**

The **Window** menu contains commands to control the arrangement of and navigation within the script windows.

**Tile Horizontal**

Use this command to arrange the script windows horizontally, one above the other.

**Tile Vertical**

Use this command to arrange the script windows vertically, one beside the other.

**Cascade**

Use this command to overlap each window in a cascading fashion.

**Arrange Icons**

Use this command to arrange minimized windows icons.

**Next Window**

Use this command to make the next window the currently selected window.

**Previous Window**

Use this command to make the previous window the currently selected window.

**Help**

The **Help** menu contains commands to display on-line help and information about the product.

**Contents**

Use this command to display the contents of the on-line help.

**This Window**

Use this command to receive on-line help for this window.

**About...**

Use this command to display copyright and product version information.

**Editor Properties**

This dialog is used to change the properties or appearance of a script window.

**Edit Window Font**

The controls in this group affect the font typeface, size and intensity used to display the script.

**Font Name**

From this drop-down list box, choose from the list of non-proportional, fixed fonts installed on your PC.

**Size**

With this spin wheel, increase or decrease the font size.

**Bold**

Check this box to increase the font intensity.

**Tab Size**

With this spin wheel, increase or decrease the number of characters between tab characters.

**Highlight Colors**

Use this group to assign colors to different parts of the script text.

**Set Text Color**

To change color, select the type of text (Normal text, Strings, etc.) and select from the Set Text Color drop-down list box to change the foreground.



**Set Background Color**

To change the background color, select from the Set Background Color drop-down list box.

**OK**

Click this button to accept the changes made and exit the dialog.

**Cancel**

Click this button to discard the changes made and exit the dialog.

**Help**

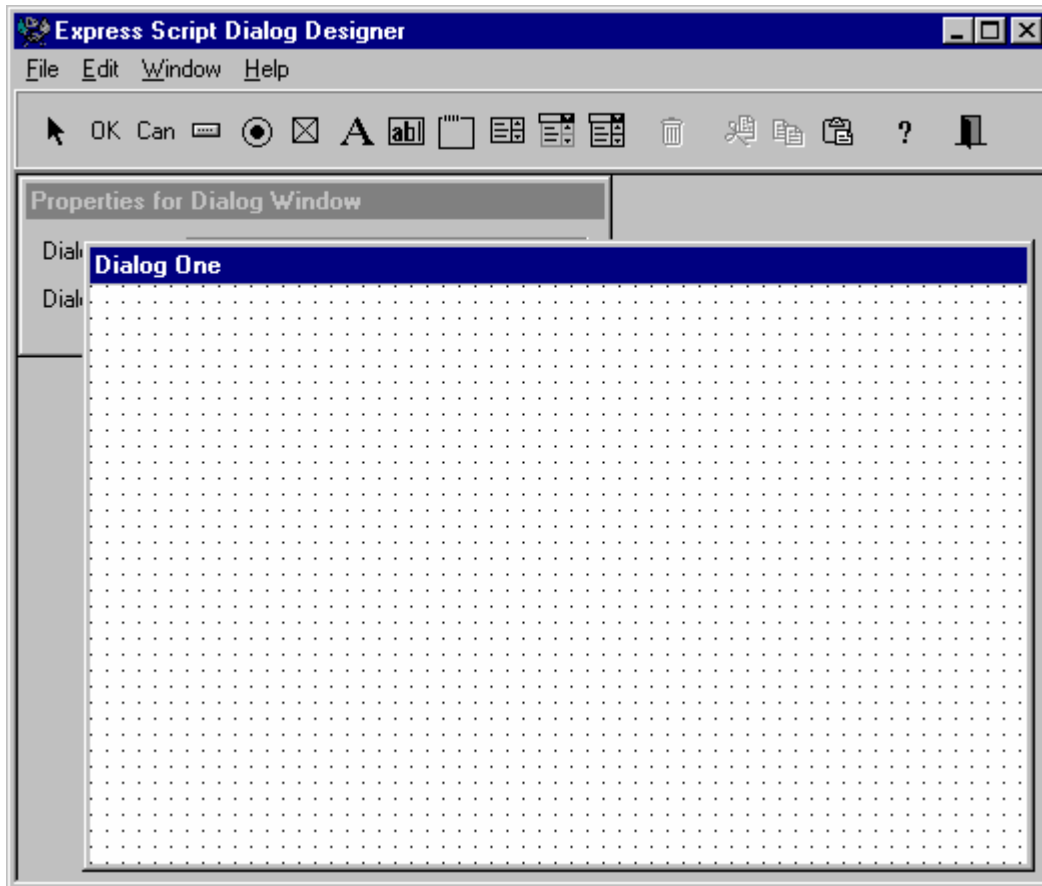
Click this button to receive on-line help for this dialog.



## The Dialog Designer

### eXpress Plus Dialog Designer

The toolbar window (titled, **Enable Dialog Designer**) contains menu items and toolbar buttons for all Enable Dialog Designer controls.



The menu items and toolbar buttons are described below. Short cut keys are shown in parentheses.

#### File Menu

The **File** menu contains commands to pass the entire dialog between the eXpress Plus Script Editor and the Dialog Designer.

##### **Load Dialog from Clipboard (F9)**

Paste the contents of the clipboard to the Dialog Designer's dialog form.

##### **Put Dialog on Clipboard (F10)**

Copy the dialog to the clipboard.

##### **Close Dialog Designer (Alt+F4)**

Close the Dialog Designer and return to the Enable Action Editor.

#### Edit Menu

The **Edit** menu contains commands to manage selected controls between the Dialog Designer's dialog form and the Windows clipboard. The menu also contains commands allowing the sizing and positioning of controls.

**Cut (Ctrl+X)**

Copy the selected controls to the clipboard and remove the control from the dialog.

**Copy (Ctrl+C)**

Copy the selected controls to the clipboard.

**Paste (Ctrl+V)**

Paste the contents of the clipboard to the dialog.

**Delete Selected (Ctrl+Del)**

Delete or clear the selected control.

**Delete All (Ctrl+A)**

Delete all controls from the dialog and reset the dialog name and title to the defaults.

**Align to Grid**

Align the selected control to the grid. This selection is only functional if the **Snap to Grid** option is set (see Grid Options, below).

**Bring to Front**

Bring the selected control to the front of the dialog display.

**Send to Back**

Send the selected control to the back of the dialog display.

**Align Controls...**

Bring up the Alignment dialog where the selected controls may be aligned vertically and/or horizontally (see Alignment).

**Size Controls...**

Bring up the Size dialog where the selected controls may be sized to match other controls (see Size).

**Grid Options...**

Bring up the Grid Options dialog where the dialog designer grid may be managed (see Grid Options).

**Window Menu****Properties Window (F2)**

Bring the Enable Dialog Designer property dialog to the front of the display.

**Design Window (F3)**

Bring the Enable Dialog Designer dialog to the front of the display.

**Help**

The **Help** menu contains commands to display on-line help and information about the product.



















**Contents**

Use this command to display the contents of the on-line help.

**About...**

Use this command to display copyright and product version information.

## Toolbar Buttons

-  Pointer mode, cancels add of a control.
-  Add OK Button (F2).
-  Add Cancel Button (F3).
-  Add Push Button (F4).
-  Add Radio Button (F5).
-  Add Check Box (F6).
-  Add Text Label (F7).
-  Add Edit or Text Box (F8).
-  Add Group Box (F9).
-  Add Standard List Box (F10).
-  Add Combo List Box (Shift+F1).
-  Add Drop-down List Box (Shift+F2).
-  Clear or Delete Selected Controls (Ctrl+Del).
-  Cut Selected Controls to Clipboard (Ctrl+X).
-  Copy Selected Controls to Clipboard (Ctrl+C).
-  Paste Contents of Clipboard to Dialog (Ctrl+V).
-  Get Help on Dialog Designer.
-  Exit Enable Dialog Designer.

## Grid Options

The Grid Options dialog allows the specification of how controls are to behave when moved and sized in the dialog, the size of the grid and whether the grid is to be displayed or hidden while in the dialog designer.

### Show Grid

If this box is checked, the dialog design grid will be displayed. The grid is only used for design purposes and may be useful when aligning dialog controls.

### Snap to Grid

This check box determines how control placement is made when controls are moved on the Dialog Designer's form. If checked, controls will snap to grid points when moved, allowing for precise alignment of controls. When not checked, no precision of alignment is attempted.

**Grid Size**

These text boxes allow you to specify spacing between grid dots both vertically and horizontally.

**Alignment**

The Alignment dialog allows any group of controls to be automatically aligned with another control.

**Horizontal**

Controls that are to be arranged above one another may be aligned to their left edges (**Left Sides**) or right edges (**Right Sides**), centered above one another (**Centers**), spaced horizontally equidistant from each other (**Space Equally**) or horizontally centered in the dialog window (**Center in window**). The **No Change** option (default) allows the vertical alignment to be changed without altering the horizontal.

**Vertical**

Controls that are to be arranged beside one another may be aligned to their top edges (**Tops**) or bottom edges (**Bottoms**), centered beside one another (**Centers**), vertically spaced equidistant from each other (**Spaced Equally**) or vertically centered in the dialog window (**Center in window**). The **No Change** option (default) allows the horizontal alignment to be changed without altering the vertical.

**Size**

The Size dialog allows any group of controls to be automatically sized to another control.

**Width**

The width of the selected controls may be reduced in size to match that of the smallest control selected (**Shrink to smallest**) or increased to match that of the largest selected (**Grow to largest**). For more precise sizing, select the **Width** option and enter a numeric value in the adjacent text box. The **No Change** option (default) allows the height to be changed without altering the width.

**Height**

The height of the selected controls may be reduced in size to match that of the smallest control selected (**Shrink to smallest**) or increased to match that of the largest selected (**Grow to largest**). For more precise sizing, select the **Height** option and enter a numeric value in the adjacent text box. The **No Change** option (default) allows the width to be changed without altering the height.

**Index**

<b>A</b>		DoTerminalKey	11
Align	25	GetScreenAttribute	11
Alignment	27	GetScreenColor	11
<b>B</b>		GetScreenLine	11
BasicLogging Property	11	OpenSession	11
<b>C</b>		OpenSessionStation	11
CancelButton	25	SetCursorPosition	11
CheckBox	25	SetScreenText	11
CloseSession Method	11	Wait	11
ComboBox	25	WaitForString	11
CursorColumn Property	11	WaitForStringNot	11
CursorRow Property	11	<b>O</b>	
<b>D</b>		OKButton	25
DatacomOptions Property	11	OpenSession Method	11
DetailLogging Property	11	OpenSessionStation Method	11
Dialog	25	OptionButton	25
DoDataKey Method	11	OptionGroup	25
DoTerminalKey Method	11	<b>P</b>	
DropListBox	25	Properties	11
<b>E</b>		BasicLogging	11
Editor Properties	23	CursorColumn	11
Express Plus32 Dialog Designer	25	CursorRow	11
<b>G</b>		DatacomOptions	11
GetScreenAttribute Method	11	DetailLogging	11
GetScreenColor Method	11	KeyboardOptions	11
GetScreenLine Method	11	LastErrorCode	11
GetScreenText Method	11	LastErrorMessage	11
Grid Options	27	SessionOpen	11
GroupBox	25	TraceOption	11
<b>H</b>		TransmitType	11
How QPlexView Works	3	Version	11
<b>K</b>		VideoOptions	11
Key Code Values	17	PushButton	25
KeyboardOptions Property	11	<b>Q</b>	
<b>L</b>		QPlexView Host Connection	9
LastErrorCode Property	11	QPlexView Overview	1
LastErrorMessage Property	11	QPlexView Script Prototype Editor/Tester dialog	21
ListBox	25	<b>R</b>	
<b>M</b>		Radio Button	25
Methods	11	<b>S</b>	
CloseSession	11	SessionOpen Property	11
DoDataKey	11		

SetCursorPosition Method	11	TransmitType Property	11
SetScreenText Method	11	<b>U</b>	
Size	25, 27	UTS Keys	17
<b>T</b>		<b>V</b>	
T27 Keys	17	Version Property	11
Text	25	VideoOptions Property	11
TextBox	25	<b>W</b>	
The ASP Script.	5	Wait Method	11
The QPlexView ActiveX Component Reference	11	WaitForString Method	11
TraceOption Property	11	WaitForStringNot Method	11