QPlexView UTS Overview

QPlexView UTS Overview

QPlexView UTS is a companion product of QPlex Client UTS, the Web-based connectivity software from KMSYS Worldwide, Inc. QPlexView UTS is an ActiveX component that provides a mechanism to Web-enable legacy applications running on Unisys mainframes.

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 "Introducing Active Server Pages" from Microsoft. On the Web go to msdn.microsoft.com and use the MSDN search engine to look for "Introducing 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.

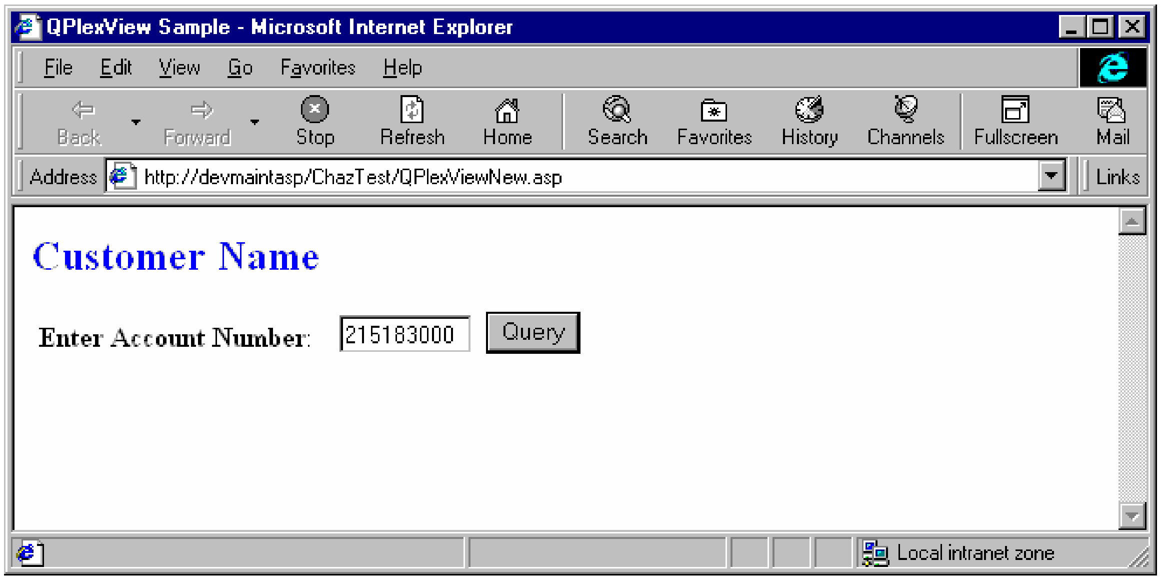
In addition to running in a client side application, QPlexView also supports all the printing capabilities of QPlex Client. The QPlex Client configuration program can be used to generate printer settings for use in QPlexView. See the ClientMode and Printer properties below. Also, see the AddPCXlate,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

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.

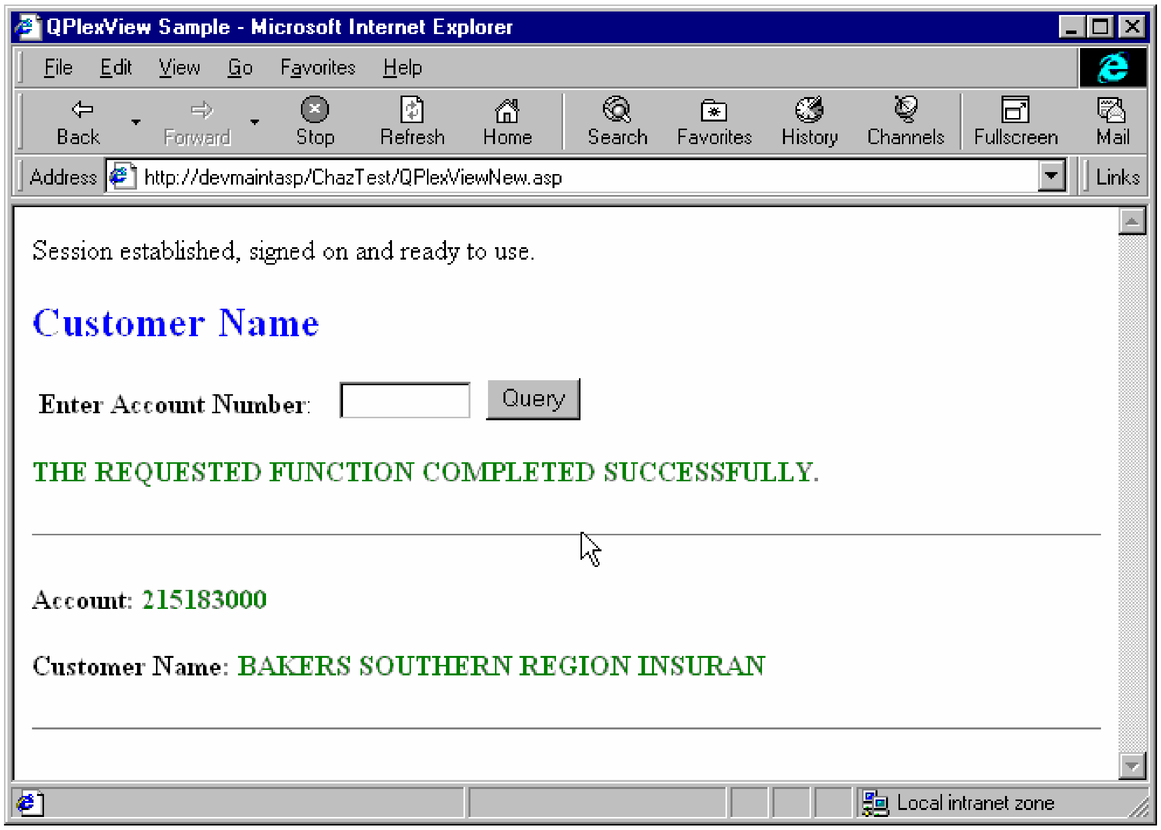

QPlexView Sample - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Address http://devmaintasp/ChazTest/QPlexViewNew.asp
Links
Query
Local intranet zone

Customer Name

Enter Account Number: 215183000

Figure 1: Initial (Blank) Page with User-Entered Account Number

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


QPlexView Sample - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Address http://devmaintasp/ChazTest/QPlexViewNew.asp
Links
Local intranet zone

Session established, signed on and ready to use.

Customer Name

Enter Account Number: Query

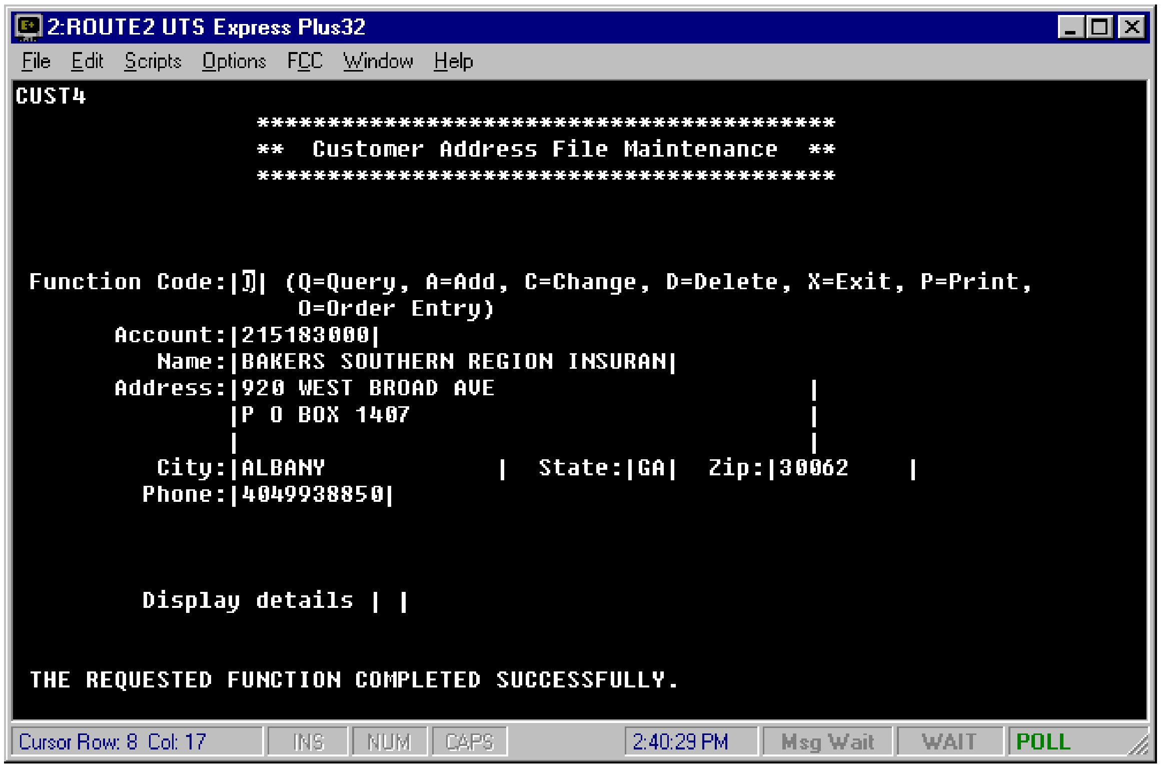
THE REQUESTED FUNCTION COMPLETED SUCCESSFULLY.

Account: 215183000

Customer Name: BAKERS SOUTHERN REGION INSURAN

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.


2:ROUTE2 UTS Express Plus32
File Edit Scripts Options FCC Window Help
Cursor Row: 8 Col: 17
INS
NUM
CAPS
2:40:29 PM
Msg Wait
WAIT
POLL

```
CUST4
                 *****************************************
                 **  Customer Address File Maintenance  **
                 *****************************************



 Function Code:|Q| (Q=Query, A=Add, C=Change, D=Delete, X=Exit, P=Print,
                    O=Order Entry)
       Account:|215183000|
          Name:|BAKERS SOUTHERN REGION INSURAN|
       Address:|920 WEST BROAD AVE                        |
               |P O BOX 1407                              |
               |                                          |
          City:|ALBANY             |  State:|GA|  Zip:|30062    |
         Phone:|4049938850|



         Display details | |


 THE REQUESTED FUNCTION COMPLETED SUCCESSFULLY.
```

Figure 3: 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

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 indciate 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 sesion 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 sesison to the 2200. See earlier ocurrance
'        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 requried,
'        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 transation 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 specfic 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 contril within the ASP script. This action could be acomplished by
'         quering the value of a button that is used to activate the page and script.
'         The action would be acomplished 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.
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;
  &nbsp;&nbsp;<!--webbot bot="Validation" startspan B-Value-Required="TRUE"
  I-Minimum-Length="9" I-Maximum-Length="9" --><!--webbot bot="Validation" endspan --><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 executed. 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.

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 Gost Gateway Server (HGS) in the same manner as QPlex Client.

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

Property	Type	Description
		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.
CursorRow	Integer Read Only	The current cursor row. If a session is not open, a 0 is returned.
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	Integer Read only	Indicates if the keyboard is locked. 1 = Keyboard locked, 0 = unlocked.
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.
Pages	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 somewhere on the screen page before utilizing the Page Up/Page Down keys. 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. WARNING: The host is totally unaware of the

Property	**Type**	**Description**
		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.
MessageWaiting	Integer Read Only	Indicates if message waiting is set. 1 = message waiting set, 0 = not set.
Printer1 Printer2 Printer3	String Read/Write	A string that specifies the configured printer settings. There are 3 different printer properties and methods (see the AddPCXlate methods on page 14) 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. All printer settings are the same for all QPlex Client Session screens. That is, you cannot configure different printers for each screen. The String values used in the PrinterX property methods are generated using the configuration program. Note: This property only available when using QPlexView in ClientMode (see ClientMode property, above).
PrintTimeOut	Integer Read/Write	-1 (or any negative value) = No timeout, 0 = Timeout immediately after printing, *n* (any positive number) = Timeout *n* seconds after printing. The default is 15 seconds.
ReadOnly	Integer Read/Write	Default is 1 (True). ReadOnly controls whether or not the end-used is allowed to type directly into the screen. If false the only way to enter anything in the screen is programmatically in the script. This property may be set when running in ClientMode and should be set before the OpenSession method is called.
ScreenFont	String Read/Write	A string that specifies the configured screen font, style and size. This property may be set when running in ClientMode and should be set before the OpenSession method is called.

Property	**Type**	**Description**
SessionOpen	Integer Read Only	Indicates whether a host session is currently open. 1 = Open, 0 = Not Open.
ShowInactivePage	Integer Read/Write	Use to show the inactive page screen. 1 = show, 0 = do not show. 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. 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). When using this option, the screen window may have to be resized manually to accommodate the dual screen display. Note: The inactive page will not appear until the user does the first Page Down or Page Up. WARNING: 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.
ShowStatusBar	Integer Write only	Use to show a status bar at the bottom of the screen. 1 = show, 0 = do not show.
SITAOpenId	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	Default is 0 (none). Set this property to 2 to a trace to file. Please see "Where to Look for the Trace

Property	Type	Description
		Files" below.
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 Write Only	Default is 0 (none). To start a Transport trace 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	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.

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 stardard 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: "KMSTransEx*nnnnnnnn*.trc" for a transport trace, "eXpressTrace*nnnnnnnn*.txt" for an emulator trace and "ETEMTrace*nnnnnnnn*.txt" for a T27 ETEM trace, where *nnnnnnnn* is 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:

Employed when using QPlexView in ClientMode, 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:

Constant	**Value**	**Description**
SATTR_NORMAL	0	Normal field
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 intensitity
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 HoldMessages 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:

Force the visible screen to be repainted. This procedure should be called after entering text into the screen or moving the cursor. Only use this procedure in ClientMode.

Send

Function **Send**(*TextToSend* as Srtring) as Integer

Description:

In ClientMode, 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
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

Code Value	Key
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