Introducing HostExplorer Programming

HostExplorer products provide a wide range of application programming interfaces (APIs), a document standard used to program applications. These APIs let you exploit the functionality and features of HostExplorer products from within your own programs and scripts.

Rather than creating your own code to redesign applications, you can use the available HostExplorer APIs. These APIs let you extend the functionality of your available programming languages (for example, Visual C++ and Visual Basic) to write scripts. With HostExplorer programming you can:

· Use OLE Automation APIs to add File Transfer capabilities to your own application.

· Use a small amount of Visual Basic script to embed an HETerminal screen within a web page, as well as add HTML user interface features for absolute control over how the terminal is used.

· Use Visual Basic or C++ scripts to provide a new front end to an existing application, and use the Parser or Terminal objects to communicate back to the modified application.

· Automate terminal display panels to improve their appearance and usability.

· Automate repetitive tasks (for example, checking data) which improves the reliability of data.

You can customize the following HostExplorer programs using the corresponding application programming interfaces (APIs) and available scripts.

· HostExplorer

· FTP

· WyseTerm

For information on creating, compiling, and debugging scripts using Hummingbird Basic Language, see the Hummingbird Basic Workbench help. Hummingbird Basic Workbench is an application that is available with the Hummingbird Accessories product. If you want to view the help file, install Hummingbird Basic Workbench, if you have not done so already.

Related Topics

Overview of HostExplorer APIs

Overview of FTP APIs

Overview of WyseTerm APIs

Introducing HostExplorer APIs

As part of the latest business trend, companies are rethinking how to access valuable information from the mainframe. Programmers need to create applications that make better use of host information. They need application programming interfaces (APIs) to allow for PC-to-host or UNIX-to-host communication.

HostExplorer provides a wide range of APIs that let you automate and use HostExplorer functionality from within your own programs and scripts.

With programming languages such as C++ and Basic, you can use the methods and properties within these APIs to customize HostExplorer to suit your needs or those of your customers. For example, you can use these APIs to:

· redesign a graphical user interface (GUI) in an application

· incorporate an application into a Web page

· create interactive Web sites

HostExplorer APIs are based on the following popular standards:

· COM (Common Object Model)

· OHIO (Open Host Interface Objects)

· OLE (Object Linking and Embedding) automation and HLLAPI (High Level Language API)

HostExplorer COM Objects

COM objects provide methods and properties that let you manipulate the behavior of objects and create relationships between objects. These objects offer the most efficient means of accessing an application's features and functionality, and they can be used by any COM-compliant application. You can use a COM object to make a direct call, and the system returns a pointer to that interface.

HostExplorer provides the following COM objects, which allow you to seamlessly integrate HostExplorer functionality within your own applications:

· Terminal objects

Note: The three Terminal objects (3270, 5250, and VT) are ActiveX objects which let you embed the HostExplorer terminal into your own applications.

· Session object

· Parser objects

· Transport objects

A functional diagram illustrates how COM objects work together to access host data from the mainframe. 

The most common libraries are:

· HostExplorer 3270 Type Library

· HostExplorer 5250 Type Library

· HostExplorer VT Type Library

· HESession 1.0 Type Library

In Visual Basic, you can add the visual controls (for example, 3270, 5250, and VT type libraries) to the project by clicking Component on the Project menu. You can add objects such as HESession and HEOhio by clicking References on the Project menu. When you add the objects, they become available in a drop-down menu. Using this drop-down menu, you can select objects in Dim statements, as well as other Visual Basic statements.

When you are using the visual controls and you add the basic object, this object is displayed on the component bar. When you add the selected control to a form in the project, Visual Basic automatically creates the object. In the following example, Visual Basic automatically creates the Session and Transport objects. Visual Basic automatically creates other objects after you connect to the session, therefore, you must assign references to these objects. In the following example, the active control is named My3270 in the project. This control is an instance of the HE3270Terminal object that you added as a visual component to the project:

Example:

```
Dim MySession As HESession
```

```
Dim MyTransport As Object

…

Set MySession = My3270.Session

Set MyTransport = My3270.Transport
```

OHIO

OHIO is a developing standard; it addresses the need for a standardized programming interface to host data. HostExplorer provides Ohio interfaces, which contain methods and properties that you can use to access different types of host data.

The Ohio object consists of classes, such as OhioManager and OhioSession, as described in the draft IETF standard. This standard is available using anonymous login at the following ftp site: ftp://ftp.boulder.ibm.com/software/standards/ohio.

In Visual Basic, you typically declare objects in one of the following formats:

· Dim OManager As HEOHIOLib.OhioManager

· Dim OSession As OhioSession

After you add the HEOhio 1.0 Type Library, the most common library for Ohio, to the project references list, either of the two formats will work. OhioManager, OhioSession, and HEOHIOLib appear in the drop-down menu that is displayed when you type the Visual Basic “AS” keyword.

Note: In Ohio, you must create the object as follows:

```
Set OManager=CreateObject(“HEOhio.OhioManager”)
```

Legacy APIs

In addition to COM objects and OHIO, HostExplorer provides the following existing, or “legacy,” APIs:

· OLE Automation—A Windows tool that lets you automate the exchange of data between applications, and lets you access and control HostExplorer.

· EHLLAPI (Extended HLLAPI) and WinHLLAPI (Windows HLLAPI)—Allow other Windows programs (for example, Attachmate® Extra! for Windows) to communicate and control HostExplorer terminal emulators.

· DDE (Dynamic Data Exchange)—A tool that allows programs (for example, Microsoft Excel, Word, and Visual Basic) to communicate with the HostExplorer 3270 emulator.

While these APIs are less efficient and use larger and more rigid objects than COM and OHIO, you can still use them to write applications and thus avoid rewriting your own code. HostExplorer’s support of these earlier APIs helps maximize an organization’s investment in its development.

Related Topics

About COM Objects

About OHIO

About Legacy APIs

About the Terminal Objects

The Terminal objects govern the terminal display and its menus. You can use the Terminal objects to create display-based applications. At the programming front end, you can also use the Terminal objects to reference the other HostExplorer API objects — the Parser objects, the Transport objects, the Session object, and Ohio.

HostExplorer consists of the following terminal emulators:

· HostExplorer TN3270—Emulates 3270 terminals for IBM mainframes.

· HostExplorer TN5250—Emulates 5250 terminals for AS/400 computers, IBM's family of mid-range computers.

· HostExplorer Telnet—Emulates ASCII terminals (VTxxx, ANSI, and SCO ANSI) for UNIX, DEC, and other ASCII-based host components.

The three terminal emulators correspond to the following Terminal objects:

· HETM3270

· HETM5250

· HETMVT

Methods of the Terminal Objects

The following are methods of the Terminal objects:

ChooseTerminalFont

EditSessionProperties

Related Topics

Properties of the Terminal Objects

Method: IHETerminal::ChooseTerminalFont 3270 5250 VT

This method generates the Font Selection dialog box.

Basic Syntax	HETerminal.**ChooseTerminalFont**
C++ Syntax	HRESULT IHETerminal::**ChooseTerminalFont**();
Parameters	This method has no parameters.
Basic Example	Dim Session As HESession Set Session = Terminal.Session Session.ProfileName = "C:\aix.hep" Session.Load Terminal.Connected = True Terminal.ChooseTerminalFont

C++ Example

```
IHESession* pISession;
IDispatch* pIDispatch;
pITerminal->get_Session(&pIDispatch);
pIDispatch->QueryInterface( IID_IHESession, (void**) &pISession);

BSTR bstrProfileName = SysAllocString(OLESTR("C:\\aix.hep"));
pISession->put_ProfileName(bstrProfileName);
pISession->Load();
SysFreeString(bstrProfileName);

VARIANT_BOOL bConnected = TRUE;
pITerminal->put_Connected(bConnected);
pITerminal->ChooseTerminalFont();
```

Method: IHETerminal::Connect 3270 5250 VT

This method establishes a connection to the host.

Basic Syntax
HETerminal.**Connect**

C++ Syntax
HRESULT IHETerminal::**Connect**();

Parameters
This method has no parameters.

Basic Example

```
Dim bVal As Boolean
bVal = Terminal.Connected
if (bVal = False) Then
Terminal.Connect();
End If
```

C++ Example

```
HRESULT Connect();
/* check if connected*/
pITerminal->get_Connected(&bConnected);
if (bConnected==VARIANT_FALSE)
{
/*not, then connect*/
pITerminal->Connect();
}
```

Method: IHETerminal::Disconnect 3270 5250 VT

This method terminates the connection to the host.

Basic Syntax HETerminal.**Disconnect**

C++ Syntax HRESULT IHETerminal::**Disconnect**();

Parameters This method has no parameters.

Basic Example

```
Dim bVal As Boolean

bVal = Terminal.Disconnected

if (bVal = True) Then

Terminal.Disconnect();

End If
```

C++ Example

```
HRESULT Disconnect();

/* check if connected*/

pITerminal->get_Connected(&bVal);

if (bVal==VARIANT_TRUE)

{

/*if connected, then disconnect*/

pITerminal->Disconnect();

}
```

Method: IHETerminal::EditSessionProperties 3270 5250 VT

This method generates the Edit Session Properties dialog box.

Basic Syntax Boolean = HETerminal.**EditSessionProperties**

C++ Syntax HRESULT IHETerminal::**EditSessionProperties**(VARIANT_BOOL * retVal);

Parameters retval—A returned value of 1 indicates the operation was successful. A returned value of 0 indicates the operation was unsuccessful.

Basic Example

```
Dim Session As HESession

Dim retval As Integer

Set Session = Terminal.Session

Session.ProfileName = "C:\aix.hep"

Session.Load

Terminal.Connected = True

Terminal.EditSessionProperties retval
```

C++ Example

```
IHESession* pISession;
IDispatch* pIDispatch;
pITerminal->get_Session(&pIDispatch);
pIDispatch->QueryInterface( IID_IHESession, (void**) &pISession);

BSTR bstrProfileName = SysAllocString(OLESTR("C:\\aix.hep"));
pISession->put_ProfileName (bstrProfileName);
pISession->Load();
SysFreeString(bstrProfileName);

VARIANT_BOOL bConnected = TRUE;
pITerminal->put_Connected(bConnected);
pITerminal->ChooseTerminalFont();

short iRetval;
pITerminal->EditSessionProperties (&iRetval);
```

Properties of the Terminal Objects

Properties define the characteristics of an object. The Terminal objects have the following properties:

Connected

Host

Session

TCPPort

Transport

Related Topics

Methods of the Terminal Objects

Property: IHETerminal::ConnectBy 3270 5250 VT

This property returns or sets a value indicating the method of connection (connection protocol) between the client machine and the host. By default, the property is set to HOSTEX_CONNECT_BY_TELNET.

Basic Syntax

HOSTEX_CONNECT_BY = HETerminal.**ConnectBy**

HETerminal.**ConnectBy** = HOSTEX_CONNECT_BY

C++ Syntax

HRESULT IHETerminal::**get_ConnectBy**([out, retval] HOSTEX_CONNECT_BY *pVal);

HRESULT IHETerminal::**put_ConnectBy**([in] HOSTEX_CONNECT_BY * newVal);

Parameters

pVal—The returned value, specifying the connection method.

newVal—The set value, specifying the connection method.

Basic Example

```
Dim ConnBy As HOSTEX_CONNECT_BY

ConnBy = Terminal.ConnectBy

If (ConnBy <> HOSTEX_CONNECT_BY_MSSNA) Then

Terminal.ConnectBy =

HOSTEX_CONNECT_BY_MSSNA

End If
```

C++ Example

```
HOSTEX_CONNECT_BY ConnBy;

Terminal.get_ConnectBy(&ConnBy);

if (ConnBy != HOSTEX_CONNECT_BY_MSSNA)

{

ConnBy = HOSTEX_CONNECT_BY_MSSNA;

Terminal.put_ConnectBy(ConnBy);

}
```

Related Topics

HOSTEX_CONNECT_BY Data Type

Property: IHETerminal::Connected 3270 5250 VT

This property returns or sets a value indicating the current connection status.

Basic Syntax

Boolean = HETerminal.**Connected**

HETerminal.**Connected** = Boolean

C++ Syntax

HRESULT IHETerminal::**get_Connected**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHETerminal::**put_Connected**([in] VARIANT_BOOL * *newVal*)

Parameters

pVal—A returned value of VARIANT_TRUE indicates that you are currently connected to the host. A returned value of VARIANT_FALSE indicates that you are currently disconnected from the host.

newVal—A value of VARIANT_TRUE indicates that you are currently connected to the host. A value of VARIANT_FALSE indicates that you are currently disconnected from the hsot.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Session.ProfileName = "C:\aix.hep"

Session.Load

‘Connect to host

Terminal.Connected = True

If (Terminal.Connected = True) Then

Terminal.Connected = False

End If
```

C++ Example

```
IHESession* pISession;

IDispatch* pIDispatch;

pITerminal->get_Session(&pIDispatch);

pIDispatch->QueryInterface( IID_IHESession, (void**) &pISession);


BSTR bstrProfileName = SysAllocString(OLESTR("C:\\aix.hep"));

pISession->put_ProfileName(bstrProfileName);

pISession->Load();

SysFreeString(bstrProfileName);


//connect

VARIANT_BOOL bConnected = TRUE;

pITerminal->put_Connected(bConnected);


//check if connected, if so then disconnect

pITerminal->get_Connected(&bConnected);

if (bConnected==TRUE)

{

//disconnect

bConnected = FALSE;

pITerminal->put_Connected(bConnected);

}
```

Property: IHETerminal::Host 3270 5250 VT

This property returns or sets the address of the host.

Basic Syntax
String = HETerminal.**Host**
HETerminal.**Host** = String

C++ Syntax
HRESULT IHETerminal::**get_Host**([out, retval] BSTR * *pVal*);
HRESULT IHETerminal::**put_Host**([in] BSTR *newVal*);

Parameters
pVal—The returned address of the current host.
newVal—The address that you set for the current host.

Basic Example

```
If (Session.HostName <> "aix “ Then

Terminal.Host = "aix"

End If



Terminal.Connected = True



'get the current Host name

Dim HostName As String

HostName = Terminal.Host

Terminal.Host = "sunset.cs.concordia.ca"
```

C++ Example

```
BSTR bstrHost = SysAllocString(OLESTR(“aix”);

pITerminal->put_Host(bstrHost);

SysFreeString(bstrHost);



bstrHost= SysAllocString(OLESTR("sunset.cs.concordia.ca"));

pITerminal->put_Host(bstrHost);
```

Property: IHETerminal::Parser 3270 5250 VT

This property returns the pointer to the currently loaded Parser.

Basic Syntax
Dispatch = HETerminal.**Parser**

C++ Syntax
HRESULT IHETerminal::**get_Parser**([out, retval] IDispatch * * *pVal*);

Parameters
pVal—The returned address of the currently loaded Parser.

Basic Example

```
Dim Parser As HEParser

Set Parser = Terminal.Parser

Parser.SendKeys "ls -alt"
```

C++ Example

```
IDispatch* pIDispatch;

IHEParser* pIParser;

PITerminal->get_Parser(&pIDispatch);

pIDispatch-> QueryInterface(IID_IHEParser,(void**)&pIParser);

BSTR bstrKeys = SysAllocString(OLESTR("ls -l"));

pIParser->SendKeys(bstrKeys);

SysFreeString(bstrKeys);
```

Property: IHETerminal::Session 3270 5250 VT

This property returns or sets the Session object which contains all the methods and properties for the session.

Basic Syntax

```
Dispatch = HETerminal.Session
HETerminal.Session = IDispatch
```

C++ Syntax

```
HRESULT IHETerminal::get_Session([out, retval] IDispatch * * pVal);
HRESULT IHETerminal::put_Session([in] IDispatch * newVal);
```

Parameters

pVal—The returned address of the Session object.
newVal—The set address of the Session object.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Session.ProfileName = "C:\aix.hep"

Session.Load

Terminal.Connected = True
```

C++ Example

```
IHESession* pISession;

IDispatch* pIDispatch;

pITerminal->get_Session(&pIDispatch);

pIDispatch->QueryInterface( IID_IHESession, (void**) &pISession);



BSTR bstrProfileName = SysAllocString(OLESTR("C:\\aix.hep"));

pISession->put_ProfileName(bstrProfileName);

pISession->Load();

SysFreeString(bstrProfileName);
```

Property: IHETerminal::SilentConnect 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer displays a progress dialog box while connecting to the host.

Basic Syntax

Boolean = HETerminal.**SilentConnect**

HETerminal.SilentConnect = Boolean

C++ Syntax

HRESULT IHETerminal::get_**SilentConnect**([out, retval] VARIANT_BOOL * p*Val*);

HRESULT IHETerminal::put_**SilentConnect**([in] VARIANT_BOOL * new*Val*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer connects your machine to the host without opening a progress dialog box. A returned value of VARIANT_FALSE indicates that a progress dialog box opens when HostExplorer connects your machine to the host.

newVal—A value of VARIANT_TRUE indicates that HostExplorer connects your machine to the host without opening a progress dialog box. A value of VARIANT_FALSE indicates that a progress dialog box opens when HostExplorer connects your machine to the host.

Basic Example

```
Dim bVal As Boolean

bVal = Terminal.SilentConnect



if (bVal = True) Then

Terminal.SilentConnect = False

End If
```

C++ Example

```
VARIANT_BOOL bVal;

Terminal->get_SilentConnect(&bVal);

if (bVal==VARIANT_TRUE)

{

bVal=VARIANT_FALSE;

Terminal->put_SilentConnect(&bVal);

}
```

Property: IHETerminal::TCPPort 3270 5250 VT

This property returns or sets the TCP/IP port for the connection.

Basic Syntax

Integer = HETerminal.**TCPPort**

HETerminal.**TCPPort** = Integer

C++ Syntax

HRESULT IHETerminal::**get_TCPPort**([out, retval] short * *pVal*);

HRESULT IHETerminal::**put_TCPPort**([in] short *newVal*);

Parameters

pVal—The returned value indicating the TCP/IP port for the connection.

newVal—The set value indicating the TCP/IP port for the connection.

Basic Example

```
If (Session.Port <>23) Then

Terminal.TCPPort = 23

End If



Terminal.Connected = True
```

```
Terminal.TCPPort = 17
```

C++ Example

```
short iPort;

pISession->get_Port(&iPort);

if (iPort != 23)

{

iPort=23;

pITerminal->put_TCPPort(iPort);

}



IPort = 17;

pITerminal->put_TCPPort (iPort);
```

Property: IHETerminal::Transport 3270 5250 VT

This property returns the address to the Transport object.

Basic Syntax Dispatch = HETerminal.**Transport**

C++ Syntax HRESULT IHETerminal.**get_Transport**([out, retval] IDispatch * * *pVal*);

Parameters *pVal*—The returned address of the Transport object.

Basic Example

```
Dim Transport As HETransport

Set Transport = Terminal.Transport

Transport.Connected = False
```

C++ Example

```
IDispatch* pIDispatch;

IHETransport* pITransport;

pITerminal->get_Transport(&pITransport);

pIDispatch-> QueryInterface(IID_IHETransport,(void**)&pITransport);

pITransport->put_Connected(FALSE);
```

Property: IHETerminal::UserDir 3270 5250 VT

This property returns or sets a value specifying where the user directory (which stores profiles, schemes, and macros) is located.

Basic Syntax

String = HETerminal.**UserDir**
HETerminal.**UserDir** = String

C++ Syntax

HRESULT IHETerminal.**get_UserDir**([out, retval] BSTR * *pVal*);
HRESULT IHETerminal.**put_UserDir**([in] BSTR * *newVal*);

Parameters

pVal—The returned value, specifying the location of the user directory.
newVal—The set value, specifying the location of the user directory.

Basic Example

```
Dim str As String

str = Terminal.UserDir



if (Len(str)=0) Then

Terminal.UserDir =

"C:\\Program Files\Hummingbird"

End If
```

C++ Example

```
BSTR bstr;

Terminal.get_UserDir(&bstr);

if (strlen( OLE2A(bstr)) == 0 )

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR(

"C:\\Program Files\Hummingbird"))

Terminal.put_UserDir(bstr);

SysFreeString(bstr);

}
```

HOSTEX_CONNECT_BY Data Type 3270 5250 VT

The HOSTEX_CONNECT_BY data type specifies the transport type that HostExplorer uses to connect to a host.

It has the following values:

HOSTEX_CONNECT_BY_TELNET	Indicates that HostExplorer connects to the host using TCP/IP.
HOSTEX_CONNECT_BY_MODEM	Indicates that HostExplorer connects to the host using a modem. This data type applies only to TNVT terminals.
HOSTEX_CONNECT_BY_MSSNA	Indicates that HostExplorer connects to the host using a Microsoft SNA server gateway. This data type applies only to TN3270 terminals.
HOSTEX_CONNECT_BY_NWSAA	Indicates that HostExplorer connects to the host using a Novell NetWare for SAA gateway. This data type applies only to TN3270 terminals.
HOSTEX_CONNECT_BY_DEMOLINK	Indicates that HostExplorer starts a demo session, which you can use to play back demo files that you previously recorded using the Dlg Save Demo File system command. This data type applies only to TN3270 terminals.

Related Topics

Property: IHEParser::ConnectBy

Property: IHESession::ConnectBy

Property: IHETerminal::ConnectBy

About the Parser Objects

The Parser objects analyze the data that is received from the Transport objects. By parsing the information from the Transport buffer, the Parser objects create a new buffer containing information that will eventually be displayed on the screen.

The Parser objects are:

· HEPAR3270—Translates information received from the 3270 data stream protocol.

· HEPAR5250—Translates information received from the 5250 data stream protocol.

· HEPARVT—Translates information received from the VT data stream protocol.

For HEPAR3270 and HEPAR5250 objects, the buffer is in EBCDIC format. For the HEPARVT object, the buffer is in ASCII format.

There are methods, properties, and/or data types specific to:

· only the HEPAR3270 object

· both the HEPAR3270 and HEPAR5250 objects

· only the HEPARVT object

There are also methods, properties, and data types common to all three objects.

Methods, Properties, and Data Types of the HEPAR3270 Object

The following methods, properties, and data types are specific to the HEPAR3270 object:

Methods

GetCecp

SetCecp

Properties

APLInputMode

DetectChainedIO

EnableAPL

EnableAutoInsertToNextField

EnableOEMReply

GraphicsCursorType

GraphicsModel

InsertKeyStyle

PassthruMode7171

PrinterDeInitString

PrinterInitString

ProgramSymbols

TPrintOutput

XferErrorCode

XferMode

Data Types

HOSTEX_GRAPHICS_CELLSIZE

HOSTEX_GRAPHICS_CURSOR_TYPE

HOSTEX_GRAPHICS_MODEL

HOSTEX_INSERT_KEY_STYLE

HOSTEX_TRANSFER_HOSTTYPE

HOSTEX_TRANSFER_INITIALACTION

HOSTEX_TRANSFER_RECORDFORMAT

Methods, Properties, and Data Types of the HEPAR3270/5250 Objects

The following methods, properties, and data types are specific to both the HEPAR3270 and the HEPAR5250 objects:

Methods

SendAid

Properties

CharSet

ConvertNulls

Cutmode

EnableAutoDeleteFromNextField

EnableAutoNextField

LeftMargin

MaxUndoRedoEvents

NextFieldKey

PasteMode

ReplaceFieldAttributeWith

TypeAheadTimeout

ValidateNumericFieldData

Data Types

HOSTEX_CELL_DELIMITED

HOSTEX_CUT_MODE

HOSTEX_FIELD_ATTR_REPLACEMENT

HOSTEX_NEXT_FIELD_KEY

HOSTEX_PASTE_MODE

HOSTEX_STATUS_LINE_MODE

HOSTEX_TRANSFER_TARGET

HOSTEX_TRANSFER_TYPE

Properties and Data Types of the HEPARVT Object

The following properties and data types are specific to the HEPARVT object:

Properties

BufRows

CaptureMode

EnablePrinterTimeout

HistoryLines

HostWritableString

Language

NRC

ScrollStart

SoftCharacterSetID

TerminalID

VTMoveCursorOnMouseClick

VTNRC

VTScrollSpeed

VTUPSS

WrapLines

Data Types

HOSTEX_CAPTURE_MODE

HOSTEX_LINEMODE

HOSTEX_TELNETECHO

HOSTEX_TERMINAL_ID

HOSTOVERWRITE_BEHAVIOUR

Methods, Properties, and Data Types of the HEPAR3270/5250/VT Objects

The following methods, properties, and data types are common to the HEPAR3270, HEPAR5250, and HEPARVT objects:

Methods

ClearSel

GetSel

ReplaceSel

SendKeys

SetCursorPosition

SetSel

Properties

BellMargin

CellDelimited

Columns

ConnectBy

EnableDisplayHostAddressOnOIA

EnableDisplayRowColumnOnOIA

Feature

HLLAPIName

KeyboardLocked

ModelColumns

ModelRows

NVTMode

OIAString

Password

Rows

SelectionMode

SessionName

StatusLineMode

TerminalModel

Transport

Data Types

HEPARSER_FEATURE

HOSTEX_ATN_FORMAT

HOSTEX_CONNECT_BY

HOSTEX_KEYBOARD_TYPE

HOSTEX_SELECTION_MODE

HOSTEX_SWITCHSCREENTYPE

HOSTEX_TERM_MODEL

Methods of the Parser Objects

The following are methods of the Parser objects:

ClearSel

GetCecp

GetSel

ReplaceSel

SendAid

SendKeys

SetCecp

SetCursorPosition

SetSel

Method: IHEParser::ClearSel 3270 5250 VT

This method lets you clear the previous selection on the terminal screen.

Basic Syntax
HEParser.**ClearSel**

C++ Syntax
HRESULT IHEParser::**ClearSel**();

Parameters
This method has no parameters.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim rc As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser =

ActiveSession.HEParserPtr

Parser.ClearSel
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->ClearSel();
```

Method: IHEParser::GetCecp 3270

This method requests four bytes to identify the terminal Country Extended Code Page (CECP) to the host.

Basic Syntax
HEParser.**GetCecp**(*cElems* As Long) As BYTE

C++ Syntax
HRESULT IHEParser::**GetCecp**([in] long *cElems*, [out, size_is(*cElems*)] BYTE * *rgs*);

Parameters
cElems—The number of characters that you receive in the string.
rgs—The array containing four bytes, identifying the language to the host.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim rc As Integer

Dim rgs As Byte

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

rc = Parser.SetCecp(4, rgs)
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

Byte * myByte;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->SetCursorPosition(4,

myByte)
```

Method: IHEParser::GetSel 3270 5250 VT

This method lets you retrieve the text in the requested selection of the SetSel method.

Basic Syntax HEParser.**GetSel**(*pData* As String)

C++ Syntax HRESULT IHEParser::**GetSel**(BSTR * *pData*);

Parameters *pData*—The returned string.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim rc As Integer

Dim sCelString As String



Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

rc = Parser.ReplaceSel(sCelString)
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

bstr sSel = “Here goes my text”;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->ReplaceSel(sSel);



SysFreeString(sSel);
```

Method: IHEParser::ReplaceSel 3270 5250 VT

This method lets you write text in the selection that you specified in the SetSel method.

Basic Syntax HEParser.**ReplaceSel**(*newText* As String)

C++ Syntax HRESULT IHEParser::**ReplaceSel**(BSTR *newText*);

Parameters *newText*—The text that you type in the selection.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim rc As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

rc = Parser.ReplaceSel(“Here goes my text”)
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

bstr sSel = “Here goes my text”;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->ReplaceSel(sSel);



SysFreeString(sSel);
```

Method: IHEParser::SendAid 3270 5250

This method sends an “aid” or special keystroke (such as the Enter key, Tab key, or Page Up key) to the Parser object.

Basic Syntax HEParser.**SendAid**(*nAidKey* As Long)

C++ Syntax HRESULT IHEParser::**SendAid**([in] long *nAidKey*);

Parameters *nAidKey*—The aid key that is sent to the Parser object.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

Parser.SendAid(65)
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->SendAid(65);
```

Related Topics

TN3270 Keyboard Mapping

TN5250 Keyboard Mapping

TN3270 Keyboard Mapping

The following list outlines the keyboard mapping for TN3270 terminals:

ALA-Alternate-Input	VK3_ALA_INPUT
Change-Graphics-Cursor	VK3_TOGGLEGRAPHICS CURSOR
Check-Hotspot	VK3_CHECKHOTSPOT
Close-Window	VK3_CLOSEWINDOW
Cursor-Select	VK3_CURSORSELECT
Dead-Key	VK3_DEAD
Dlg-API-Settings	IDM_O_EHLLAPI
Dlg-Close-Session	IDM_F_CLOSESESSION
Dlg-Download	IDM_T_RECEIVEDOWNLOAD
Dlg-Edit-Session-Profile	IDM_O_PREFERENCES
Dlg-Exit	IDM_F_EXIT
Dlg-Font-Select	IDM_N_SELECTFONT
Dlg-Global	IDM_O_GLOBAL
Dlg-Hotspots	IDM_O_HOTSPOTS
Dlg-Keyboard-Mapper	IDM_O_KEYMAPR
Dlg-Open-Session	IDM_F_OPENSESSION
Dlg-Poppad-Configure	IDM_O_POPPAD

Dlg-Poppad-Custom	IDM_V_CUSTOM_POPPAD
Dlg-Poppad-Default	IDM_V_DEFAULT_POPPAD
Dlg-Print-Screen	IDM_F_PRINTSCREEN
Dlg-Prompt-Demo-File	VK3_ASKFORDEMOFILE
Dlg-Prompt-Password	VK3_PROMPTFORPASSWORD
Dlg-Quick-Key-Editor	IDM_O_MACRO
Dlg-Run-Macro	IDM_M_MACRORUN
Dlg-Run-Program	VK3_RUNPGMDLG
Dlg-Save-Demo-File	VK3_SAVEDEMOFILE
Dlg-Save-Profile	IDM_F_SAVESESSION
Dlg-Save-Screen	IDM_F_SAVESCREEN
Dlg-Translate-Tables	IDM_O_XLATEDLG
Dlg-Upload	IDM_T_SENDUPLOAD
End-Recording	IDM_M_MACROENDRECORDING
Font-Larger	IDM_N_NEXTLARGERFONT
Font-Smaller	IDM_N_NEXTSMALLERFONT
Help-Index	IDM_INDEXHELP
Help-Keys	IDM_KEYHELP
Jump-To-Session	VK3_JUMPSESSION
Insert-Field-Attribute	VK3_INSERTFIELDATTRIBUTE
IPause	VK3_IPAUSE
Maximize-Font	IDM_N_MAXIMIZEFONT
Mouse-To-Cursor	VK3_MOUSETOCURSOR
Move-Cursor-Cursor-Select	VK3_MOVECURSORAND CURSORSELECT
Move-Cursor-Enter	VK3_MOVECURSORANDENTER
Next-Session	VK3_NEXT_SESSION
Password	VK3_PASSWORD
Pause	VK3_PAUSE
Prev-Session	VK3_PREV_SESSION
Print-Raw	VK3_PRINTRAW
Print-Raw-LPT1	VK3_PRINTRAWLPT1
Print-Raw-LPT2	VK3_PRINTRAWLPT2
Print-Raw-LPT3	VK3_PRINTRAWLPT3
Print-Screen	VK3_PRINTSCREEN
Record-Macro	IDM_M_MACRORECORD
Reset-Type-Ahead	VK3_RESET_TYPEAHEAD
Restore-Cursor-Position	VK3_RESTORECURSOR POSITION
Run	VK3_RUNPGM
Run-Macro	VK3_RUNSCRIPT
Save-Cursor-Position	VK3_SAVECURSORPOSITION
Save-Screen	VK3_SAVESCREEN
Show-Demo-File	VK3_SHOWDEMOFILE
Show-Track-Menu	VK3_SHOWTRACKMENU
Start-Session	VK3_STARTSESSION
Toggle-APL-Keyboard	VK3_APLTOGGLE
Toggle-Attribute	VK3_FLIP_ATTR
Toggle-Capture	IDM_F_CAPTURE
Toggle-Connection	VK3_TOGGLE_CONNECTION
Toggle-CrossHair-Cursor	VK3_TOGGLECROSSHAIR
Toggle-Cursor	VK3_TOGGLE_CURSOR
Toggle-Entry-Assist	VK3_ENTRY_ASSIST
Toggle-Full-Screen	VK3_FULLSCREEN
Toggle-Recording-Pause	IDM_M_MACROPAUSE RECORDING
Toggle-Toolbar	IDM_V_TOOLBAR
Toggle-Tracing	VK3_TOGGLETRACE
Toggle-Word-Wrap	VK3_WORD_WRAP
Whats-This?	VK3_WHATSTHIS

TN5250 Keyboard Mapping

The following keystrokes are available for TN5250 terminals:

Check-Hotspot	VK3_CHECKHOTSPOT
Close-Window	VK3_CLOSEWINDOW
Cursor-Select	VK3_CURSORSELECT
Dead-Key	VK3_DEAD
Dlg-API-Settings	IDM_O_EHLLAPI
Dlg-Close-Session	IDM_F_CLOSESESSION
Dlg-Download	IDM_T_RECEIVEDOWNLOAD
Dlg-Edit-Session-Profile	IDM_O_PREFERENCES
Dlg-Exit	IDM_F_EXIT
Dlg-Font-Select	IDM_N_SELECTFONT
Dlg-Global	IDM_O_GLOBAL
Dlg-Hotspots	IDM_O_HOTSPOTS
Dlg-Keyboard-Mapper	IDM_O_KEYMAPR
Dlg-Open-Session	IDM_F_OPENSESSION
Dlg-Poppad-Configure	IDM_O_POPPAD
Dlg-Poppad-Custom	IDM_V_CUSTOM_POPPAD
Dlg-Poppad-Default	IDM_V_DEFAULT_POPPAD
Dlg-Print-Screen	IDM_F_PRINTSCREEN
Dlg-Prompt-Demo-File	VK3_ASKFORDEMOFILE
Dlg-Prompt-Password	VK3_PROMPTFORPASSWORD
Dlg-Quick-Key-Editor	IDM_O_MACRO
Dlg-Run-Macro	IDM_M_MACRORUN
Dlg-Run-Program	VK3_RUNPGMDLG
Dlg-Save-Demo-File	VK3_SAVEDEMOFILE
Dlg-Save-Profile	IDM_F_SAVESESSION
Dlg-Save-Screen	IDM_F_SAVESCREEN
Dlg-Translate-Tables	IDM_O_XLATEDLG
Dlg-Upload	IDM_T_SENDUPLOAD
End-Recording	IDM_M_MACROEND RECORDING
Font-Larger	IDM_N_NEXTLARGERFONT
Font-Smaller	IDM_N_NEXTSMALLERFONT
Help-Index	IDM_INDEXHELP
Help-Keys	IDM_KEYHELP
Insert-Field-Attribute	VK3_INSERTFIELDATTRIBUTE
IPause	VK3_IPAUSE
Jump-To-Session	VK3_JUMPSESSION
Maximize-Font	IDM_N_MAXIMIZEFONT
Mouse-To-Cursor	VK3_MOUSETOCURSOR
Move-Cursor-Cursor-Select	VK3_MOVECURSORAND CURSORSELECT
Move-Cursor-Enter	VK3_MOVECURSORANDENTER
Next-Session	VK3_NEXT_SESSION
Password	VK3_PASSWORD
Pause	VK3_PAUSE
Prev-Session	VK3_PREV_SESSION
Print-Raw	VK3_PRINTRAW
Print-Raw-LPT1	VK3_PRINTRAWLPT1
Print-Raw-LPT2	VK3_PRINTRAWLPT2
Print-Raw-LPT3	VK3_PRINTRAWLPT3
Print-Screen	VK3_PRINTSCREEN
Record-Macro	IDM_M_MACRORECORD
Reset-Type-Ahead	VK3_RESET_TYPEAHEAD
Restore-Cursor-Position	VK3_RESTORECURSOR POSITION
Run	VK3_RUNPGM
Run-Macro	VK3_RUNSCRIPT
Save-Cursor-Position	VK3_SAVECURSORPOSITION

Save-Screen	VK3_SAVESCREEN
Show-Demo-File	VK3_SHOWDEMOFILE
Show-Track-Menu	VK3_SHOWTRACKMENU
Start-Session	VK3_STARTSESSION
Toggle-Attribute	VK3_FLIP_ATTR
Toggle-Capture	IDM_F_CAPTURE
Toggle-Connection	VK3_TOGGLE_CONNECTION
Toggle-CrossHair-Cursor	VK3_TOGGLECROSSHAIR
Toggle-Cursor	VK3_TOGGLE_CURSOR
Toggle-Entry-Assist	VK3_ENTRY_ASSIST
Toggle-Full-Screen	VK3_FULLSCREEN
Toggle-Recording-Pause	IDM_M_MACROPAUSE RECORDING
Toggle-Toolbar	IDM_V_TOOLBAR
Toggle-Tracing	VK3_TOGGLETRACE
Toggle-Word-Wrap	VK3_WORD_WRAP
Whats-This?	VK3_WHATSTHIS

Method: IHEParser::SendKeys 3270 5250 VT

This method sends general keystrokes (other than function keys) to the host.

Basic Syntax HEParser.**SendKeys**(*pBuffer* As String)

C++ Syntax HRESULT IHEParser::**SendKeys**(BSTR *pBuffer*);

Parameters *pBuffer*—The string containing the keystrokes being sent.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

Parser.SendKeys ("Hello")
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())
```

pHEParser->SendKeys(“Hello”);

Method: IHEParser::SetCecp 3270

This method sets the four bytes required to identify the terminal Country Extended Code Page (CECP) to the host.

Basic Syntax HEParser.**SetCecp**(*cElems* As Long, *rgs* As BYTE)

C++ Syntax HRESULT IHEParser::**SetCecp**([in] long *cElems*, [in, size_is(*cElems*)] BYTE * *rgs*);

Parameters *cElems*—The number of characters in the string.
rgs—The array containing four bytes, identifying the language to the host.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim rc As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

rc = Parser.SetCecp(4,4)
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->SetCecp(4, (BYTE *)4)
```

Method: IHEParser::SetCursorPosition 3270 5250 VT

This method lets you specify the row and column position of the cursor.

Basic Syntax HEParser.**SetCursorPosition**(*Row* As Integer, *Column* As Integer)

C++ Syntax HRESULT IHEParser::**SetCursorPosition**(short *Row*, short *Column*);

Parameters *Row*—The row position of the cursor.
Column—The column position of the cursor.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim rc As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

rc = Parser.SetCursorPosition(5, 5)
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->SetCursorPosition(5,5)
```

Method: IHEParser::SetSel 3270 5250 VT

This method creates a selection using the following four coordinates: start row, end row, start column, and end column.

Basic Syntax HEParser.**SetSel**(*StartRow* As Integer, *StartCol* As Integer, *EndRow* As Integer, *EndCol* As Integer)

C++ Syntax HRESULT IHEParser::**SetSel**(short *StartRow*, short *StartCol*, short *EndRow*, short *EndCol*);

Parameters *StartRow*—The first row of the selection.
StartCol—The first column of the selection.
EndRow—The last row of the selection

EndCol—The last column of the selection.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim rc As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

rc = Parser.SetSel(1,1,5,5)
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{

pHEParser->SetSel(1,1,5,5);

}
```

Related Topics

Property: IHEParser::SelectionMode

Properties of the Parser Objects

Properties define the characteristics of an object. The Parser objects have the following properties:

APLInputMode	NextFieldKey
BellMargin	NRC
BufRows	NVTMode
CaptureMode	OIAString
CellDelimited	PassthruMode7171
CharSet	Password
Columns	PasteMode
ConnectBy	PrinterDeInitString

Property: IHEParser::APLInputMode 3270

This property indicates whether the session is in APL (A Program Language) input mode. By default, this property is set to VARIANT_FALSE.

Basic Syntax Boolean = HEParser.**APLInputMode**

C++ Syntax HRESULT IHEParser::**get_APLInputMode**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the session is in APL input mode. A returned value of VARIANT_FALSE indicates that the session is not in APL input mode.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim bInputMode As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

bInputModel = Parser.APLInputMode
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL pVal;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

pHEParser->get_ APLInputMode(&pVal);
```

Property: IHEParser::BellMargin 3270 5250 VT

This property creates a beeping sound when your cursor reaches the specified column of an input field. By default, this property is set to 0.

Basic Syntax

Integer = HEParser.**BellMargin**
HEParser.**BellMargin** = Integer

C++ Syntax

HRESULT IHEParser::**get_BellMargin**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_BellMargin**([in] short *newVal*);

Parameters

pVal—The returned value, indicating the specified column.
newVal—The set value, indicating the specified column.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iBellMargin As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iBellMargin = Parser.BellMargin

'Set the value

Parser.BellMargin = 3
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

Short pVal =0, newVal = 3;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_BellMargin(&pVal);

// set value

pHEParser->put_BellMargin(newVal);

}
```

Property: IHEParser::BufRows VT

This property returns or sets the maximum number of rows, including the scrollback buffer. This number can vary between 1 and 9,999. By default, this property is set to 100. To disable the scrollback buffer, set the number to 0.

Basic Syntax

Integer = HEParser.**BufRows**

HEParser.**BufRows** = Integer

C++ Syntax

HRESULT IHEParser::**get_BufRows**([out, retval] short * *pVal*);

HRESULT IHEParser::**put_BufRows**([in] short *newVal*);

Parameters

pVal—The returned maximum number of rows.

newVal—The maximum number of rows that you set.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim BufRows As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

BufRows = Parser.BufRows

'Set the value

Parser.BufRows = 300
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

Short pVal =0, newVal = 300;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_BufRows(&pVal);

// set value

pHEParser->get_BufRows(newVal);

}
```

Property: IHEParser::CaptureMode VT

This property specifies how to capture selected text.

Basic Syntax

HOSTEX_CAPTURE_MODE = HEParser.**CaptureMode**

HEParser.**CaptureMode** = HOSTEX_CAPTURE_MODE

C++ Syntax

HRESULT IHEParser::**get_CaptureMode**([out, retval] HOSTEX_CAPTURE_MODE * *pVal*);

HRESULT IHEParser::**put_CaptureMode**([in] HOSTEX_CAPTURE_MODE *newVal*);

Parameters

pVal—The returned value, indicating whether the capture mode is raw or binary.

newVal—The set value, indicating whether the capture mode is raw or binary.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_CAPTURE_MODE_NEWVAL As

HOSTEX_CAPTURE_MODE



Set OLE1 = CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_CAPTURE_MODE_NEWVAL =

Parser.CaptureMode

'Set the value

Parser.CaptureMode =

HOSTEX_CAPTURE_MODE_TEXT
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOST_EX_CAPTURE_MODE HOSTEX_CAPTURE_MODE_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())
```

```
{ // get value

pHEParser->get_CaptureMode

(&HOSTEX_CAPTURE_MODE_VAL);

// set value

pHEParser->put_CaptureMode

(HOSTEX_CAPTURE_MODE_TEXT);

}
```

Related Topics

HOSTEX_CAPTURE_MODE Data Type

Property: IHEParser::CellDelimited 3270 5250 VT

This property enables CSV and BIFF formats when HostExplorer copies data to the clipboard and pastes data from other applications. When copying data to the clipboard in cell-delimited format, HostExplorer can parse screen data at words or at field attributes. This lets you determine how data appears in cells in your spreadsheet application.

Basic Syntax

HOSTEX_CELL_DELIMITED = HEParser.**CellDelimited**
HEParser.**CellDelimited** = HOSTEX_CELL_DELIMITED

C++ Syntax

HRESULT IHEParser::**get_CellDelimited**([out, retval] HOSTEX_CELL_DELIMITED * *pVal*);
HRESULT IHEParser::**put_CellDelimited**([in] HOSTEX_CELL_DELIMITED *newVal*);

Parameters

pVal—The returned value which indicates how data appears in cells.
newVal—The set value which indicates how data appears in cells.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_CELL_DELIMITED_VAL As

HOSTEX_CELL_DELIMITED

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_CELL_DELIMITED_VAL =

Parser.CellDelimited

'Set the value

Parser.CellDelimited =

HOSTEX_CELL_DELIMITED_FIELD
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_CELL_DELIMITED HOSTEX_CELL_DELIMITED_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_ CellDelimited( &

HOSTEX_CELL_DELIMITED _VAL);

// set value

pHEParser->put_CellDelimited

(HOSTEX_CELL_DELIMITED_FIELD);

}
```

Related Topics

HOSTEX_CELL_DELIMITED Data Type

Property: IHEParser::CharSet 3270 5250

This property returns or sets a language index of which translation table to use in the HETranslate object that is loaded in the Parser objects. For TN3270 terminals, the default is 0x0216. For TN5250 terminals, the default is 0x0113.

Basic Syntax

```
Word = HEParser.CharSet
HEParser.CharSet = Word
```

C++ Syntax

```
HRESULT IHEParser::get_CharSet([out, retval] WORD * pVal);
HRESULT IHEParser::put_CharSet([in] WORD newVal);
```

Parameters

pVal—The returned index of the translation table used by the HETranslate object for the language.

newVal—The value that you set for the index of the translation table used by the HETranslate object for the language.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim Word As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr



'get the value

Word = Parser.Charset

'Set the value

Parser.Charset = 275
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

Word wCharset, newCharSet = 0x0113;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_CharSet(&wCharSet);

// set value

pHEParser->put_CharSet(newCharSet);

}
```

Related Topics

TN3270 Language-Conversion Table

TN5250 Language-Conversion Table

Property: IHEParser::Columns 3270 5250 VT

This property returns the number of columns that the terminal supports.

Basic Syntax
Integer = HEParser.**Columns**

C++ Syntax
HRESULT IHEParser::**get_Columns**([out, retval] short * *pVal*);

Parameters
pVal—The returned number of columns that is set by the terminal model.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iColumns As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

iColumns = Parser.Columns
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

Short iColumns;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);

pHEParser->get_Columns(&iColumns);
```

Property: IHEParser::ConnectBy 3270 5250 VT

This property returns or sets a value indicating the method of connection (connection protocol) between the client machine and the host. By default, the property is set to HOSTEX_CONNECT_BY_TELNET.

Basic Syntax
HOSTEX_CONNECT_BY = HEParser.**ConnectBy**
HEParser.**ConnectBy** = HOSTEX_CONNECT_BY

C++ Syntax
HRESULT IHEParser::**get_ConnectBy**([out, retval] HOSTEX_CONNECT_BY * *pVal*);
HRESULT IHEParser::**put_ConnectBy**([in] HOSTEX_CONNECT_BY *newVal*);

Parameters
pVal—The returned value, indicating the connection.
newVal—The value that you set, indicating the connection.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_CONNECT_BY_VAL As

HOSTEX_CONNECT_BY

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr



'get the value

HOSTEX_CONNECT_BY_VAL =

HEParser.ConnectBy

'Set the value

HEParser.ConnectBy =

HOSTEX_CONNECT_BY_DEMOLINK
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);

if (pHEParser())

{ // get value

pHEParser->get_ConnectBy

(&HOSTEX_CONNECT_BY_VAL);

// set value

pHEParser->put_ConnectBy

(HOSTEX_CONNECT_BY_DEMOLINK);

}
```

Related Topics

HOSTEX_CONNECT_BY data type

Property: IHEParser::ConvertNulls 3270 5250

This property converts zeros (nulls) to blank characters in input fields, when you copy text from one input field to another. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**ConvertNulls**

HEParser.**ConvertNulls** = Boolean

C++ Syntax

HRESULT IHEParser::**get_ConvertNulls**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_ConvertNulls**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the system converts nulls to blank characters in input fields. A returned value of VARIANT_FALSE indicates that the system does not convert nulls to blank characters in input fields.

newVal—A value of VARIANT_TRUE indicates that the system converts nulls to blank characters in input fields. A value of VARIANT_FALSE indicates that the system does not convert nulls to blank characters in input fields.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim bNulls As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

bNulls = Parser.ConvertNulls

'Set the value

Parser.ConvertNulls = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL bConvertNulls;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);

if (pHEParser())

{ // get value

pHEParser->get_ConvertNulls

(&bConvertNulls);
```

```
// set value

pHEParser->put_ConvertNulls(TRUE);

}
```

Property: IHEParser::Cutmode 3270 5250

This property removes selected text from unprotected areas of the screen. By default, this property is set to HOSTEX_CUT_MODE_DELETE_TEXT.

Basic Syntax

HOSTEX_CUT_MODE = HEParser.**CutMode**
HEParser.**CutMode** = HOSTEX_CUT_MODE

C++ Syntax

HRESULT IHEParser::**get_CutMode**([out, retval] HOSTEX_CUT_MODE * *pVal*);
HRESULT IHEParser::**put_CutMode**([in] HOSTEX_CUT_MODE *newVal*);

Parameters

pVal—The returned value indicates how the selected text is removed.
newVal—The set value indicates how the selected text is removed.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_CUT_MODE_VAL As

HOSTEX_CUT_MODE

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_CUT_MODE_VAL = Parser.CutMode

'Set the value

Parser.CutMode =

HOSTEX_CUT_MODE_DELETE_TEXT
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_CUT_MODE HOSTEX_CUT_MODE_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);
```

```
if (pHEParser())

{ // get value

pHEParser->get_CutMode

(&HOSTEX_CUT_MODE_VAL);

// set value

pHEParser->put_CutMode

(HOSTEX_CUT_MODE_DELETE_TEXT);

}
```

Related Topics

HOSTEX_CUT_MODE Data Type

Property: IHEParser::DetectChainedIO 3270

This property detects chained I/O operations. If you encounter problems entering data on the mainframe, turn this feature off .

Basic Syntax

Boolean = HEParser.**DetectChainedIO**

HEParser.**DetectChainedIO** = Boolean

C++ Syntax

HRESULT IHEParser::**get_DetectChainedIO**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_DetectChainedIO**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer detects chained I/O operations. A returned value of VARIANT_FALSE indicates that HostExplorer does not detect chained I/O operations.

newVal—A value of VARIANT_TRUE indicates that HostExplorer detects chained I/O operations. A value of VARIANT_FALSE indicates that HostExplorer does not detect chained I/O operations.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim bChainedIO As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

bChainedIO = Parser.DetectChainedIO

'Set the value

Parser.DetectChainedIO = False
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL bChainedIO;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_DetectChainedIO

(&bChainedIO);

// set value

pHEParser->put_DetectChainedIO

(FALSE);

}
```

Property: IHEParser::EnableAPL 3270

This property enables APL (A Program Language) mode. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**EnableAPL**

HEParser.**EnableAPL** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnableAPL**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnableAPL**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the system enables APL mode. A returned value of VARIANT_FALSE indicates that the system does not enable APL mode.

newVal—A set value of VARIANT_TRUE indicates that the system enables APL mode. A set value of VARIANT_FALSE indicates that the system does not enable APL mode.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim APL As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

APL = Parser.EnableAPL

'Set the value

Parser.EnableAPL = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL APL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_EnableAPL (&APL);

// set value

pHEParser->put_EnableAPL (TRUE);

}
```

Property: IHEParser::EnableAutoDeleteFromNextField 3270 5250

This property determines whether the Delete and Backspace keys remove characters in the current and subsequent lines. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**EnableAutoDeleteFromNextField**

HEParser.**EnableAutoDeleteFromNextField** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnableAutoDeleteFromNextField**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnableAutoDeleteFromNextField**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Delete and Backspace keys remove characters in the current and subsequent lines. A returned value of VARIANT_FALSE indicates that the Delete and Backspace keys remove characters only in the current line.

newVal—A value of VARIANT_TRUE indicates that the Delete and Backspace keys remove characters in the current and subsequent lines. A value of VARIANT_FALSE indicates that the Delete and Backspace keys remove characters only in the current line.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim ADFNF As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

ADFNF =

Parser.EnableAutoDeleteFromNextField

'Set the value

Parser.EnableAutoDeleteFromNextField =

True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL ADFNF;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())
```

```
{ // get value

pHEParser->get_EnableAutoDeleteFromNextField(&ADFNF);

// set value

pHEParser->

put_EnableAutoDeleteFromNextField

(TRUE);

}
```

Property: IHEParser::EnableAutoInsertToNextField 3270

This property determines whether the Insert key inserts characters on the current and subsequent lines. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**EnableAutoDeleteFromNextField**

HEParser.**EnableAutoDeleteFromNextField** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnableAutoInsertToNextField**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnableAutoInsertToNextField**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Insert key inserts characters and does not stop when the cursor reaches the end of the current line. A returned value of VARIANT_FALSE indicates that the Insert key inserts characters and stops when the cursor reaches the end of the current line.

newVal—A value of VARIANT_TRUE indicates that the Insert key inserts characters and does not stop when the cursor reaches the end of the current line. A value of VARIANT_FALSE indicates that the Insert key inserts characters and stops when the cursor reaches the end of the current line.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim AIFNF As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

AIFNF =

Parser.EnableAutoInsertFromNextField

'Set the value

Parser.EnableAutoInsertFromNextField =

True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL AIFNF;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_EnableAutoInsertFromNextField(&AIFNF);

// set value

pHEParser->

put_EnableAutoInsertFromNextField

(TRUE);

}
```

Property: IHEParser::EnableAutoNextField 3270 5250

When you reach the end of a field, this property lets you wrap text to the next available field. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**EnableAutoNextField**

HEParser.**EnableAutoNextField** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnableAutoNextField**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnableAutoNextField**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the data entry continues to the next input field. A returned value of VARIANT_FALSE indicates that the cursor stops at the end of the field; you must press the Tab key to continue to the next field.

newVal—A value of VARIANT_TRUE indicates that the data entry continues to the next input field. A value of FALSE indicates that the cursor stops at the end of the field; you must press the Tab key to continue to the next field.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim EANF As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

EANF = Parser.EnableAutoNextField

'Set the value

Parser.EnableAutoNextField = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL EANF;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_EnableAutoNextField

(&EANF);

// set value

pHEParser->put_EnableAutoNextField

(TRUE);

}
```

Property: IHEParser::EnableDisplayHostAddressOnOIA 3270 5250 VT

This property indicates whether to display the host address in the Operator Information Area (OIA).

Basic Syntax

Boolean = HEParser.**EnableDisplayHostAddressOnOIA**

HEParser.**EnableDisplayHostAddressOnOIA** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnableDisplayHostAddressOnOIA**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnableDisplayHostAddressOnOIA**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the host address in the OIA is displayed. A returned value of VARIANT_FALSE indicates that the host address in the OIA is not displayed.

newVal—A set value of VARIANT_TRUE indicates that the host address in the OIA is displayed. A set value of VARIANT_FALSE indicates that the host address in the OIA is not displayed.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HostAdrOIA As Boolean

Set OLE1 = CreateObject

("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HostAdrOIA =

Parser.EnableDisplayHostAddressOnOIA

'Set the value

Parser.EnableDisplayHostAddressOnOIA =

True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL HostAdrOIA;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())
```

```
{ // get value

pHEParser->

get_EnableDisplayHostAddressOnOIA

(&HostAdrOIA);

// set value

pHEParser->

put_EnableDisplayHostAddressOnOIA

(TRUE);

}
```

Property: IHEParser::EnableDisplayRowColumnOnOIA 3270 5250 VT

This property indicates whether to display the row and column coordinates in the OIA (Operator Information Area). By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = HEParser.**EnableDisplayRowColumnOnOIA**

HEParser.**EnableDisplayRowColumnOnOIA** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnableDisplayRowColumnOnOIA**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnableDisplayRowColumnOnOIA**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the row and column coordinates in the OIA are displayed. A returned value of VARIANT_FALSE indicates that the row and column coordinates in the OIA are not displayed.

newVal—A set value of VARIANT_TRUE indicates that the row and column coordinates in the OIA are displayed. A set value of VARIANT_FALSE indicates that the row and column coordinates in the OIA are not displayed.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim RCOIA As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

RCOIA =

Parser.EnableDisplayRowColumnOnOIA

'Set the value

Parser.EnableDisplayRowColumnOnOIA =

True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL RCOIA;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_EnableDisplayRowColumnOnOIA (&RCOIA);

// set value

pHEParser->

put_EnableDisplayRowColumnOnOIA

(TRUE);

}
```

Property: IHEParser::EnableOEMReply 3270

This property lets you determine whether HostExplorer sends an OEM reply field back to the host in response to receiving a Read Partition Query. If sent, the OEM reply would contain information about the terminal session and features available for the host to use. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = HEParser.**EnableOEMReply**

HEParser.**EnableOEMReply** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnableOEMReply**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnableOEMReply**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer sends an OEM reply field back to the host. A returned value of VARIANT_FALSE indicates that HostExplorer does not send an OEM reply field back to the host.

newVal—A set value of VARIANT_TRUE indicates that HostExplorer sends an OEM reply field back to the host. A returned value of VARIANT_FALSE indicates that HostExplorer does not send an OEM reply field back to the host.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim OEM As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

OEM = Parser.EnableOEMReply

'Set the value

Parser.EnableOEMReply = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL OEM;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_EnableOEMReply

(&OEM);

// set value

pHEParser->put_EnableOEMReply

(TRUE);

}
```

Property: IHEParser::EnablePrinterTimeout 3270 5250 VT

This property lets you enable or disable the delay of printer output.

Basic Syntax

Boolean = HEParser.**EnablePrinterTimeout**

HEParser.**EnablePrinterTimeout** = Boolean

C++ Syntax

HRESULT IHEParser::**get_EnablePrinterTimeout**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_EnablePrinterTimeout**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables the delay of printer output. A returned value of VARIANT_FALSE indicates that HostExplorer disables the delay of printer output.

newVal—A set value of VARIANT_TRUE indicates that HostExplorer enables the delay of printer output. A set value of VARIANT_FALSE indicates that HostExplorer disables the delay of printer output.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim PrintTimeout As Boolean

Set OLE1 = CreateObject

("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

PrintTimeout =

Parser.EnablePrinterTimeout

'Set the value

Parser.EnablePrinterTimeout = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL PrintTimeout;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_EnablePrinterTimeout
```

```
(&PrintTimeout);

// set value

pHEParser->put_EnablePrinterTimeout

(TRUE);

}
```

Property: IHEParser::Feature 3270 5250 VT

This property lets you set or reset multiple Boolean parameters in the Parser objects using one function call by specifying the parameter name (feature type) and a value (VARIANT_TRUE or VARIANT_FALSE).

Basic Syntax

Boolean = HEParser.**Feature**

HEParser.**Feature** = Boolean

C++ Syntax

HRESULT IHEParser::**get_Feature**(HEPARSER_FEATURE 1Type, [out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_Feature**(HEPARSER_FEATURE lType, [in] VARIANT_BOOL *newVal*);

Parameters

lType—The variable name that is set or reset.

pVal—A returned value of VARIANT_TRUE indicates that the specified Parser feature is enabled. A returned value of VARIANT_FALSE indicates that the Parser feature is disabled.

newVal—A value of VARIANT_TRUE indicates that the specified Parser feature is enabled. A value of VARIANT_FALSE indicates that the Parser feature is disabled.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim Feature As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

Feature = Parser.Feature

'Set the value

Parser.Feature = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL OEM;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_Feature (&OEM);

// set value

pHEParser->put_Feature (TRUE);

}
```

Related Topics

HEPARSER_FEATURE Data Type

Property: IHEParser::GraphicsCursorType 3270

This property lets you determine how the cursor appears in the terminal window.

Basic Syntax

HOSTEX_GRAPHICS_CURSOR_TYPE = HEParser.**GraphicsCursorType**

HEParser.**GraphicsCursorType** = HOSTEX_GRAPHICS_CURSOR_TYPE

C++ Syntax

HRESULT IHEParser::**get_GraphicsCursorType**([out, retval] HOSTEX_GRAPHICS_CURSOR_TYPE) * *pVal*);

HRESULT IHEParser::**put_GraphicsCursorType**([in] HOSTEX_GRAPHICS_CURSOR_TYPE) *newVal*);

Parameters

pVal—The returned value, which indicates how the cursor appears in the terminal window.

newVal—The set value, which indicates how the cursor appears in the terminal window.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_GRAPHICS_CURSOR_TYPE_VAL As HOSTEX_GRAPHICS_CURSOR_TYPE

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_GRAPHICS_CURSOR_TYPE_VAL =

Parser.GraphicsCursorType

'Set the value

Parser.GraphicsCursorType =

HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_GRAPHICS_CURSOR_TYPE HOSTEX_GRAPHICS_CURSOR_TYPE_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_GraphicsCursorType

(&HOSTEX_GRAPHICS_CURSOR_TYPE_VAL);

// set value

pHEParser->put_GraphicsCursorType

(HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE);

}
```

Related Topics

HOSTEX_GRAPHICS_CURSOR_TYPE Data Type

Property: IHEParser::GraphicsModel 3270

This property lets you select the graphics terminal model to use during the next session.

Basic Syntax

HOSTEX_GRAPHICS_MODEL = HEParser.**GraphicsModel**

HEParser.**GraphicsModel** = HOSTEX_GRAPHICS_MODEL

C++ Syntax

HRESULT IHEParser::**get_GraphicsModel**([out, retval] HOSTEX_GRAPHICS_MODEL * *pVal*);

HRESULT IHEParser::**put_GraphicsModel**([in] HOSTEX_GRAPHICS_MODEL *newVal*);

Parameters

pVal—The returned value, which indicates the graphics terminal model to use during the session.

newVal—The set value, which indicates the graphics terminal model to use during the session.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_GRAPHICS_MODEL_VAL As

HOSTEX_GRAPHICS_MODEL

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_GRAPHICS_MODEL_VAL =

Parser.GraphicsModel

'Set the value

Parser.GraphicsModel =

HOSTEX_GRAPHICS_MODEL_3179G
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_GRAPHICS_MODEL HOSTEX_GRAPHICS_MODEL_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);
```

```
if (pHEParser())

{ // get value

pHEParser->get_GraphicsModel

(&HOSTEX_GRAPHICS_MODEL_VAL);

// set value

pHEParser->put_GraphicsModel

(HOSTEX_GRAPHICS_MODEL_3179G);

}
```

Related Topics

HOSTEX_GRAPHICS_MODEL Data Type

Property: IHEParser::HistoryLines VT

This property returns or sets the number of lines available in the Scrollback buffer.

Basic Syntax

Integer = HEParser.**HistoryLines**
HEParser.**HistoryLines** = Integer

C++ Syntax

HRESULT IHEParser::**get_HistoryLines**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_HistoryLines**([in] short *newVal*);

Parameters

pVal—The returned number of lines available in the Scrollback buffer.
newVal—The set number of lines available in the the Scrollback buffer.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iHistory As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iHistory = Parser.HistoryLines

'Set the value

Parser.HistoryLines = 200
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

Short iHistory;


Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);


if (pHEParser())

{ // get value

pHEParser->get_HistoryLines

(&iHistory);

// set value

pHEParser->put_HistoryLines

(200);

}
```

Property: IHEParser::HLLAPIName 3270 5250 VT

This property returns or sets the HLLAPI name that provides access to a session.

Basic Syntax

String = HEParser.**HLLAPIName**
HEParser.**HLLAPIName** = String

C++ Syntax

HRESULT IHEParser::**get_HLLAPIName**([out, retval] BSTR * *pVal*);
HRESULT IHEParser::**put_HLLAPIName**([in] BSTR *newVal*);

Parameters

pVal—The returned HLLAPI name.
newVal—The HLLAPI name that you set.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HLLAPIName As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HLLAPIName = Parser.HLLAPIName



'Set the value

Parser.HLLAPIName = “A”
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR bstrHLLAPIName;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_HLLAPIName (&bstrHLLAPIName);

// set value

pHEParser->put_HLLAPIName (“A”);

}

SysFreeString(bstrHLLAPIName);
```

Property: IHEParser::HostWritableString VT

This property returns the text in the host status line of the operator information area (OiA).

Basic Syntax

String = HEParser.**HostWritableString**

C++ Syntax

HRESULT IHEParser::**get_HostWritableString**([out, retval] BSTR * *pVal*);

Parameters

pVal—The returned string that appears in the host status line of the OIA.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim sWriteString As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

sWriteString =

Parser.HostWritableString
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR bstrWriteString;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_HostWritableString

(&bstrWriteString);

}

SysFreeString(bstrWriteString);
```

Property: IHEParser::InsertKeyStyle 3270

This property lets you determine when the Insert key is toggled on. By default, this property is set to HOSTEX_INSERT_KEY_STYLE_RESET.

Basic Syntax

HOSTEX_INSERT_KEY_STYLE = HEParser.**InsertKeyStyle**

HEParser.**InsertKeyStyle** = HOSTEX_INSERT_KEY_STYLE

C++ Syntax

HRESULT IHEParser::**get_InsertKeyStyle**([out, retval] HOSTEX_INSERT_KEY_STYLE * *pVal*);

HRESULT IHEParser::**put_InsertKeyStyle**([in] HOSTEX_INSERT_KEY_STYLE *newVal*);

Parameters

pVal—The returned value, which indicates how the Insert key option operates.

newVal—The set value, which indicates how the Insert key option operates.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_INSERT_KEY_STYLE_VAL As

HOSTEX_INSERT_KEY_STYLE

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_INSERT_KEY_STYLE_VAL =

Parser.InsertKeyStyle

'Set the value

Parser.InsertKeyStyle =

HOSTEX_INSERT_KEY_STYLE_ACTION
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_INSERT_KEY_STYLE HOSTEX_INSERT_KEY_STYLE_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_InsertKeyStyle
```

```
(&HOSTEX_INSERT_KEY_STYLE_VAL);

// set value

pHEParser->put_InsertKeyStyle

(HOSTEX_INSERT_KEY_STYLE_ACTION);

}
```

Related Topics

HOSTEX_INSERT_KEY_STYLE Data Type

Property: IHEParser::KeyboardLocked 3270 5250 VT

This property locks the keyboard, and prevents HostExplorer from accepting and displaying pressed keys.

Basic Syntax

```
Integer = HEParser.KeyboardLocked
HEParser.KeyboardLocked = Integer
```

C++ Syntax

```
HRESULT IHEParser::get_KeyboardLocked([out, retval] short * pVal);
HRESULT IHEParser::put_KeyboardLocked([in] short newVal);
```

Parameters

pVal—The returned value, which indicates that the keyboard is locked.
newVal—The set value, which indicates that the keyboard is locked.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iLocked As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

‘get the value

iLocked = Parser.KeyboardLocked

‘Set the value

Parser.KeyboardLocked = 1
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iLocked;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_KeyboardLocked

(&iLocked);

// set value

pHEParser->put_KeyboardLocked(1);

}
```

Property: IHEParser::Language VT

This property returns or sets the language that you want to use.

Basic Syntax

String = HEParser.**Language**
HEParser.**Language** = String

C++ Syntax

HRESULT IHEParser::**get_Language**([out, retval] BSTR * *pVal*);
HRESULT IHEParser::**put_Language**([in] BSTR *newVal*);

Parameters

pVal—The returned string which indicates the language.
newVal—The set string which indicates the language.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim sLanguage As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

‘get the value

sLanguage = Parser.Language

‘Set the value

Parser.Language=“ISO Latin-1 (8859-1)”
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR bstrLanguage;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_Language (&bstrLanguage);

// set value

pHEParser->put_Language

(“ISO Latin-1 (8859-1)”);

}

SysFreeString(bstrLanguage);
```

Related Topics

TNVT UPSS Language-Conversion Table

TNVT NRC Language-Conversion Table

Property: IHEParser::LeftMargin 3270 5250

This property returns or sets a value for the left margin of the screen. By default, this property is set to 0.

Basic Syntax

Integer = HEParser.**LeftMargin**
HEParser.**LeftMargin** = Integer

C++ Syntax

HRESULT IHEParser::**get_LeftMargin**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_LeftMargin**([in] short *newVal*);

Parameters

pVal—The returned value for the left margin of the screen.
newVal—The set value for the left margin of the screen.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iMargin As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iMargin = Parser.LeftMargin

'Set the value

Parser.LeftMargin = 1
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iMargin;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_LeftMargin

(&iMargin);

// set value
```

```
pHEParser->put_LeftMargin (1);

}
```

Property: IHEParser::MaxUndoRedoEvents 3270 5250

This property returns or sets the maximum number of events that can be recorded. By default, this property is set to 0.

Basic Syntax

Integer = HEParser.**MaxUndoRedoEvents**
HEParser.**MaxUndoRedoEvents** = Integer

C++ Syntax

HRESULT IHEParser::**get_MaxUndoRedoEvents**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_MaxUndoRedoEvents**([in] short *newVal*);

Parameters

pVal—The returned maximum number of events that can be recorded.
newVal—The set maximum number of events that can be recorded.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iUndo As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iUndo = Parser.MaxUndoRedoEvents

'Set the value

Parser.MaxUndoRedoEvents = 20
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iUndo;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value
```

pHEParser->get_MaxUndoRedoEvents

(&iUndo);

// set value

pHEParser->put_MaxUndoRedoEvents

(20);

}

Property: IHEParser::ModelColumns 3270 5250 VT

This property returns or sets the number of columns supported by the terminal model.

Basic Syntax

Integer = HEParser.**ModelColumns**
HEParser.**ModelColumns** = Integer

C++ Syntax

HRESULT IHEParser::**get_ModelColumns**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_ModelColumns**([in] short *newVal*);

Parameters

pVal—The returned number of columns supported by the terminal model.
newVal—The set number of columns supported by the terminal model.

Basic Example

Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iCols As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iCols = Parser.ModelColumns

'Set the value

Parser.ModelColumns = 90

C++ Example

HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iCols;

Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);

```
if (pHEParser())

{ // get value

pHEParser->get_ModelColumns

(&iCols);

// set value

pHEParser->put_ModelColumns (90);

}
```

Property: IHEParser::ModelRows 3270 5250 VT

This property returns or sets the number of rows supported by the terminal model.

Basic Syntax

Integer = HEParser.**ModelRows**
HEParser.**ModelRows** = Integer

C++ Syntax

HRESULT IHEParser::**get_ModelRows**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_ModelRows**([in] short *newVal*);

Parameters

pVal—The returned number of rows supported by the terminal model.
newVal—The set number of rows supported by the terminal model.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iRows As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iRows = Parser.ModelRows

'Set the value

Parser.ModelRows = 30
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iRows;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_ModelRows

(&iRows);

// set value

pHEParser->put_ModelRows (30);

}
```

Property: IHEParser::NextFieldKey 3270 5250

This property specifies which key (Tab, Comma, or Paragraph Mark) you can use to tab to the next field.

Basic Syntax

HOSTEX_NEXT_FIELD_KEY = HEParser.**NextFieldKey**

HEParser.**NextFieldKey** = HOSTEX_NEXT_FIELD_KEY

C++ Syntax

HRESULT IHEParser::**get_NextFieldKey**([out, retval] HOSTEX_NEXT_FIELD_KEY * *pVal*);

HRESULT IHEParser::**put_NextFieldKey**([in] HOSTEX_NEXT_FIELD_KEY *newVal*);

Parameters

pVal—The returned value indicating how HostExplorer tabs to the next field.

newVal—The set value indicating how HostExplorer tabs to the next field.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_NEXT_FIELD_KEY_VAL As

HOSTEX_NEXT_FIELD_KEY

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_NEXT_FIELD_KEY_VAL =

Parser.NextFieldKey

'Set the value

Parser.NextFieldKey =

HOSTEX_NEXT_FIELD_KEY_TAB
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_NEXT_FIELD_KEY HOSTEX_NEXT_FIELD_KEY_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_NextFieldKey

(&HOSTEX_NEXT_FIELD_KEY_VAL);

// set value

pHEParser->put_NextFieldKey

(HOSTEX_NEXT_FIELD_KEY_TAB);

}
```

Related Topics

HOSTEX_NEXT_FIELD_KEY Data Type

Property: IHEParser::NRC VT

This property returns or sets the NRC (National Replacement Character) set.

Basic Syntax

String = HEParser.**NRC**
HEParser.**NRC** = String

C++ Syntax

HRESULT IHEParser::**get_NRC**([out, retval] BSTR * *pVal*);
HRESULT IHEParser::**put_NRC**([in] BSTR *newVal*);

Parameters

pVal—The returned string for the NRC set.
newVal—The set string for the NRC set.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim sNRC As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

sNRC = Parser.NRC

'Set the value

Parser.NRC = “&”
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR sNRC;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_NRC (&sNRC);

// set value
```

pHEParser->put_NRC (“&”);

}

SysFreeString(sNRC);

Related Topics

Property: IHEParser::Language

TNVT NRC Language-Conversion Table

Property: IHEParser::NVTMode 3270 5250 VT

This property indicates whether the system is in NVT mode (or Linemode).

Basic Syntax
Boolean = HEParser.**NVTMode**

C++ Syntax
HRESULT IHEParser::**get_NVTMode**([out, retval] VARIANT_BOOL * *pVal*);

Parameters
pVal—A returned value of VARIANT_TRUE indicates that the system is in NVT mode. A returned value of VARIANT_FALSE indicates that the system is not in NVT mode.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim NVT As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser =

ActiveSession.HEParserPtr

'get the value

NVT = Parser.NVTMode
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL NVT;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);
```

```
if (pHEParser())

{ // get value

pHEParser->get_NVTMode (&NVT);

}
```

Property: IHEParser::OIAString 3270 5250 VT

This property returns the text of the host status line in the OIA (Operator Information Area).

Basic Syntax

String = HEParser.**OIAString**

C++ Syntax

HRESULT IHEParser::**get_OIAString**([out, retval] BSTR * *pVal*);

Parameters

pVal—The returned text of the host status line in the OIA.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim sOIA As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser =

ActiveSession.HEParserPtr

'get the value

sOIA = Parser.OIAString
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR sOIA;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value
```

```
pHEParser->get_OIAString

(&sOIA);

}

SysFreeString(sOIA);
```

Property: IHEParser::PassthruMode7171 3270

This property specifies for what escape sequences HostExplorer searches to enable or disable the printer port.

Basic Syntax

Integer = HEParser.**PassthruMode7171**

HEParser.**PassthruMode7171** = Integer

C++ Syntax

HRESULT IHEParser::**get_PassthruMode7171**([out, retval] int * *pVal*);

HRESULT IHEParser::**put_PassthruMode7171**([in] int *newVal*);

Parameters

pVal—The returned value, indicating the escape sequences to enable or disable the printer port.

newVal—The set value, indicating the escape sequences to enable or disable the printer port.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iPass As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser =

ActiveSession.HEParserPtr

'get the value

iPass = Parser.PassthruMode7171

'Set the value

Parser.PassthruMode7171 = 1234
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iPass;


Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);


if (pHEParser())

{ // get value

pHEParser->get_PassthruMode7171

(&iPass);

// set value

pHEParser->put_PassthruMode7171

(1234);

}
```

Property: IHEParser::Password 3270 5250 VT

This property returns or sets a session password which HostExplorer displays and saves only in encrypted format.

Basic Syntax

String = HEParser.**Password**
HEParser.**Password** = String

C++ Syntax

HRESULT IHEParser::**get_Password**([out, retval] BSTR * *pVal*);
HRESULT IHEParser::**put_Password**([in] BSTR *newVal*);

Parameters

pVal—The returned string, indicating the session password.
newVal—The set string, indicating the session password.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim sPassword As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser =

ActiveSession.HEParserPtr

'get the value

sPassword = Parser.Password

'Set the value

Parser.Password = “MonkeySee”
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR sPassword;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_Password

(&sPassword);

// set value

pHEParser->put_Password

(“MonkeyDo”);

}

SysFreeString(sPassword);
```

Property: IHEParser::PasteMode 3270 5250

This property lets you determine how HostExplorer pastes the contents of the clipboard to the current cursor location. By default, this property is set to HOSTEX_PASTE_MODE_PASTE_BLOCK.

Basic Syntax

HOSTEX_PASTE_MODE = HEParser.**PasteMode**

HEParser.**PasteMode** = HOSTEX_PASTE_MODE

C++ Syntax

HRESULT IHEParser::**get_PasteMode**([out, retval] HOSTEX_PASTE_MODE * *pVal*);

HRESULT IHEParser::**put_PasteMode**([in] HOSTEX_PASTE_MODE *newVal*);

Parameters

pVal—The returned string, indicating how HostExplorer pastes the contents of the clipboard to the current cursor location.

newVal—The set string, indicating how HostExplorer pastes the contents of the clipboard to the current cursor location.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_PASTE_MODE_KEY_VAL As

HOSTEX_PASTE_MODE

Set OLE1 = CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_PASTE_MODE_KEY_VAL =

Parser.PasteMode

'Set the value

Parser.PasteMode =

HOSTEX_PASTE_MODE_PASTE_OVERLAY
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_PASTE_MODE_KEY HOSTEX_PASTE_MODE_KEY_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_PasteMode
```

```
(&HOSTEX_PASTE_MODE_KEY_VAL);

// set value

pHEParser->put_PasteMode

(HOSTEX_PASTE_MODE_PASTE_OVERLAY);

}
```

Related Topics

HOSETX_PASTE_MODE Data Type

Property: IHEParser::PrinterDeInitString 3270

This property sends the de-initialization strings to the printer for passthru printing. It also defines the escape sequences that can be sent to the printer at the end of a print job. Each string can contain up to 255 characters.

Basic Syntax

String = HEParser.**PrinterDeInitString**
HEParser.**PrinterDeInitString** = String

C++ Syntax

HRESULT IHEParser::**get_PrinterDeInitString**([out, retval] BSTR * *pVal*);
HRESULT IHEParser::**put_PrinterDeInitString**([in] BSTR *newVal*);

Parameters

pVal—The returned de-initialization strings that are sent to the printer for passthru printing.
newVal—The set de-initialization strings that are sent to the printer for passthru printing.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim sDeInit As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

sDeInit = Parser.PrinterDeInitString

'Set the value

Parser.PrinterDeInitString = “@#$%!”
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR sDeInit;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_PrinterDeInitString

(&sDeInit);

// set value

pHEParser->put_PrinterDeInitString

("@#$%!");

}

SysFreeString(sDeInit);
```

Property: IHEParser::PrinterInitString 3270

This property sends the initialization strings to the printer for passthru printing. It also defines the escape sequences that can be sent to the printer at the beginning of a print job. Each string can contain up to 255 characters.

Basic Syntax

String = HEParser.**PrinterInitString**
HEParser.**PrinterInitString** = String

C++ Syntax

HRESULT IHEParser::**get_PrinterInitString**([out, retval] BSTR * *pVal*);
HRESULT IHEParser::**put_PrinterInitString**([in] BSTR *newVal*);

Parameters

pVal—The returned initialization strings that are sent to the printer for passthru printing.
newVal—The set initialization strings that are sent to the printer for passthru printing.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim sInit As String

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

sInit = Parser.PrinterInitString

'Set the value

Parser.PrinterInitString = “@#$%!”
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR sInit;


Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);


if (pHEParser())

{ // get value

pHEParser->get_PrinterInitString

(&sInit);

// set value

pHEParser->put_PrinterInitString

(“@#$%!”);

}

SysFreeString(sInit);
```

Property: IHEParser::PrinterTimeout VT

This property returns or sets the delay (in seconds) before the printer outputs a page.

Basic Syntax

Long = HEParser.**PrinterTimeout**
HEParser.**PrinterTimeout** = Long

C++ Syntax

HRESULT IHEParser::**get_PrinterTimeout**([out, retval] long * *pVal*);
HRESULT IHEParser::**put_PrinterTimeout**([in] long *newVal*);

Parameters

pVal—The returned value, indicating the delay before the printer outputs a page.
newVal—The set value, indicating the delay before the printer outputs a page.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim printTimeout As Long

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

printTimeout = Parser.PrinterTimeout

'Set the value

Parser.PrinterTimeout = 10
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

long printTimeout;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_PrinterTimeout

(&printTimeout);

// set value

pHEParser->put_PrinterTimeout (10);

}
```

Property: IHEParser::ProgramSymbols 3270

This property lets you determine whether HostExplorer supports program symbols. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**ProgramSymbols**

HEParser.**ProgramSymbols** = Boolean

C++ Syntax

HRESULT IHEParser::**get_ProgramSymbols**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_ProgramSymbols**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer supports Program symbols. A returned value of VARIANT_FALSE indicates that HostExplorer does not support Program symbols.

newVal—A set value of VARIANT_TRUE indicates that HostExplorer supports Program symbols. A set value of VARIANT_FALSE indicates that HostExplorer does not support Program symbols.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim bSymbol As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

bSymbol = Parser.ProgramSymbols

'Set the value

Parser.ProgramSymbols = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL bSymbol;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_ProgramSymbols
```

```
(&bSymbol);

// set value

pHEParser->put_ProgramSymbols

(TRUE);

}
```

Property: IHEParser::ReplaceFieldAttributeWith 3270 5250

This property replaces field attributes with a tab, comma, or paragraph mark. By default, this property is set to HOSTEX_FIELD_ATTR_REPLACEMENT_COMMA.

Basic Syntax

HOSTEX_FIELD_ATTR_REPLACEMENT = HEParser.**ReplaceFieldAttibuteWith**

HEParser.**ReplaceFieldAttibuteWith** = HOSTEX_FIELD_ATTR_REPLACEMENT

C++ Syntax

HRESULT IHEParser::**get_ReplaceFieldAttibuteWith**([out, retval] HOSTEX_FIELD_ATTR_REPLACEMENT * *pVal*);

HRESULT IHEParser::**put_ReplaceFieldAttibuteWith**([in] HOSTEX_FIELD_ATTR_REPLACEMENT *newVal*);

Parameters

pVal—The returned value, indicating how the field attribute is replaced.

newVal—The set value, indicating how the field attribute is replaced.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_FIELD_ATTR_REPLACEMENT_VAL

As HOSTEX_FIELD_ATTR_REPLACEMENT

Set OLE1 = CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_FIELD_ATTR_REPLACEMENT_VAL =

Parser.ReplaceFieldAttibuteWith

'Set the value

Parser.ReplaceFieldAttibuteWith =

HOSTEX_FIELD_ATTR_REPLACEMENT_COMMA
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_PASTE_MODE_KEY HOSTEX_PASTE_MODE_KEY_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_ReplaceFieldAttibuteWith

(&HOSTEX_FIELD_ATTR_REPLACEMENT_VAL);

// set value

pHEParser->put_ReplaceFieldAttibuteWith

(HOSTEX_FIELD_ATTR_REPLACEMENT_COMMA);

}
```

Related Topics

HOSTEX_FIELD_ATTR_REPLACEMENT Data Type

Property: IHEParser::RightMargin 3270 5250

This property returns or sets a value for the right margin of the screen. By default, this property is set to 0.

Basic Syntax

Integer = HEParser.**RightMargin**
HEParser.**RightMargin** = Integer

C++ Syntax

HRESULT IHEParser::**get_RightMargin**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_RightMargin**([in] short *newVal*);

Parameters

pVal—The returned value for the left margin of the screen.
newVal—The set value for the right margin of the screen.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iMargin As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iMargin = Parser.RightMargin

'Set the value

Parser.RightMargin = 3
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iPass;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_RightMargin

(&iMargin);

// set value

pHEParser->put_RightMargin (3);

}
```

Property: IHEParser::Rows 3270 5250 VT

This property returns the number of rows that the terminal supports.

Basic Syntax

Integer = HEParser.**Rows**

C++ Syntax

HRESULT IHEParser::**get_Rows**([out, retval] short * *pVal*);

Parameters

pVal—The returned number of rows that the terminal supports.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iRows As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iRows = Parser.Rows

'Set the value

Parser.Rows = 30
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iRows;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_Rows (&iRows);

// set value

pHEParser->put_Rows (30);

}
```

Property: IHEParser::ScrollStart VT

This property returns the line number where the viewable area of the screen starts.

Basic Syntax
Integer = HEParser.**ScrollStart**

C++ Syntax
HRESULT IHEParser::**get_ScrollStart**([out, retval] short * **pVal**);

Parameters
pVal—The returned line number where the viewable area of the screen starts.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iScroll As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iScroll = Parser.ScrollStart

'Set the value

Parser.ScrollStar = 10
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iScroll;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_ScrollStart

(&iScroll);

// set value

pHEParser->put_ScrollStart (10);

}
```

Property: IHEParser::SelectionMode 3270 5250 VT

This property returns or sets the type of selection (that is, stream or rectangular). By default, this property is set to HOSTEX_SELECTION_MODE_BLOCK.

Basic Syntax

HOSTEX_SELECTION_MODE = HEParser.**SelectionMode**

HEParser.**SelectionMode** = HOSTEX_SELECTION_MODE

C++ Syntax

HRESULT IHEParser::**get_SelectionMode**([out, retval] HOSTEX_SELECTION_MODE * *pVal*);

HRESULT IHEParser::**put_SelectionMode**([in] HOSTEX_SELECTION_MODE *newVal*);

Parameters

pVal—The returned value, indicating the selection type.

newVal—The set value, indicating the selection type.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_SELECTION_MODE_VAL As

HOSTEX_SELECTION_MODE

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_SELECTION_MODE_VAL =

Parser.SelectionMode

'Set the value

Parser.SelectionMode =

HOSTEX_SELECTION_MODE_BLOCK
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_SELECTION_MODE HOSTEX_SELECTION_MODE_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())
```

```
{ // get value
pHEParser->get_SelectionMode
(&HOSTEX_SELECTION_MODE_VAL);
// set value
pHEParser->put_SelectionMode
(HOSTEX_SELECTION_MODE_BLOCK);
}
```

Related Topics

HOSTEX_SELECTION_MODE Data Type

Property: IHEParser::SessionName 3270 5250 VT

This property returns or sets 16 bytes of data in the Operator Information Area (OIA).

Basic Syntax

String = HEParser.**SessionName**
HEParser.**SessionName** = String

C++ Syntax

HRESULT IHEParser::**get_SessionName**([out, retval] BSTR * *pVal*);
HRESULT IHEParser::**put_SessionName**([in] BSTR *newVal*);

Parameters

pVal—The returned string that is displayed in the OIA.
newVal—The set string that is displayed in the OIA.

Basic Example

```
Dim OLE1 As Object
Dim ActiveSession As Object
Dim Parser As Object
Dim sName As String
Set OLE1 =
CreateObject("HostExplorer")
Set ActiveSession = OLE1.CurrentHost
Set Parser = ActiveSession.HEParserPtr
'get the value
sName = Parser.SessionName
'Set the value
Parser.SessionName = “A - Sess”
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BSTR sName;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_SessionName (&sName);

// set value

pHEParser->put_SessionName

(“A – Sess”);

}

SysFreeString(sName);
```

Property: IHEParser::SoftCharacterSetID VT

This property returns the ID of the soft character set that is currently loaded on the host machine.

Basic Syntax Integer = HEParser.**SoftCharacterSetID**

C++ Syntax HRESULT IHEParser::**get_SoftCharacterSetID**([out, retval] short * *pVal*);

Parameters *pVal*—The returned ID of the soft character set.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iChrID As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iChrID = Parser.SoftCharacterSetID
```

```
'Set the value

Parser.SoftCharacterSetID = 1
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iChrID;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_SoftCharacterSetID

(&iChrID);

// set value

pHEParser->put_SoftCharacterSetID

(1);

}
```

Property: IHEParser::StatusLineMode 3270 5250 VT

This property returns or sets the type of status linemode (that is, terminal or window). By default, this property is set to HOSTEX_STATUS_LINE_MODE_WINDOWSTATUSBAR.

Basic Syntax

HOSTEX_STATUS_LINE_MODE = HEParser.**StatusLineMode**
HEParser.**StatusLineMode** = HOSTEX_STATUS_LINE_MODE

C++ Syntax

HRESULT IHEParser::**get_StatusLineMode**([out, retval] HOSTEX_STATUS_LINE_MODE * *pVal*);
HRESULT IHEParser::**put_StatusLineMode**([in] HOSTEX_STATUS_LINE_MODE *newVal*);

Parameters

pVal—The returned value, indicating the type of status linemode.
newVal—The set value, indicating the type of status linemode.

Basic Example

```
Dim OLE1 As Object
Dim ActiveSession As Object
Dim Parser As Object
Dim HOSTEX_STATUS_LINE_MODE_VAL As
HOSTEX_STATUS_LINE_MODE
Set OLE1 =
CreateObject("HostExplorer")
Set ActiveSession = OLE1.CurrentHost
Set Parser = ActiveSession.HEParserPtr
'get the value
HOSTEX_STATUS_LINE_MODE_VAL =
Parser.StatusLineMode
'Set the value
Parser.StatusLineMode =
HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE
```

C++ Example

```
HRESULT Hr;
IHEParser * pHEParser = NULL;
IHostExHost * pHost = NULL;
IHostExApplication * pApplication = NULL;
HOSTEX_STATUS_LINE_MODE HOSTEX_STATUS_LINE_MODE_VAL;

Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);
pApplication->get_CurrentHost(pHost);
pHost->get_HEParserPtr(&pHEParser);

if (pHEParser())
{ // get value
pHEParser->get_StatusLineMode
(&HOSTEX_STATUS_LINE_MODE_VAL);
// set value
pHEParser->put_StatusLineMode
(HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE);
}
```

Related Topics

HOSTEX_STATUS_LINE_MODE Data Type

Property: IHEParser::TerminalID VT

This property returns or sets the terminal ID response that HostExplorer sends to the host. By default, this property is set to HOSTEX_TERMINAL_ID_VT220.

Basic Syntax

HOSTEX_TERMINAL_ID = HEParser.**TerminalID**

HEParser.**TerminalID** = HOSTEX_TERMINAL_ID

C++ Syntax

HRESULT IHEParser::**get_TerminalID**([out, retval] HOSTEX_TERMINAL_ID * *pVal*);

HRESULT IHEParser::**put_TerminalID**([in] HOSTEX_TERMINAL_ID *newVal*);

Parameters

pVal—The returned terminal ID response.

newVal—The set terminal ID response.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_TERMINAL_ID_VAL As

HOSTEX_TERMINAL_ID

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_TERMINAL_ID_VAL =

Parser.TerminalID

'Set the value

Parser.TerminalID =

HOSTEX_TERMINAL_ID_VT220
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_TERMINAL_ID HOSTEX_TERMINAL_ID_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);
```

```
if (pHEParser())
{ // get value
pHEParser->get_TerminalID
(&HOSTEX_TERMINAL_ID_VAL);
// set value
pHEParser->put_TerminalID
(HOSTEX_TERMINAL_ID_VT220);
}
```

Related Topics

HOSTEX_TERMINAL_ID Data Type

Property: IHEParser::TerminalModel 3270 5250 VT

This property returns or sets the terminal model supported by HostExplorer. For TN3270 and TN5250 terminals, the default for this property is set to HOSTEX_TERM_MODEL_2. For TNVT terminals, the default is set to HOSTEX_TERM_MODEL_VT220.

Basic Syntax

HOSTEX_TERM_MODEL = HEParser.**TerminalModel**
HEParser.**TerminalModel** = HOSTEX_TERM_MODEL

C++ Syntax

HRESULT IHEParser::**get_TerminalModel**([out, retval] HOSTEX_TERM_MODEL * *pVal*);
HRESULT IHEParser::**put_TerminalModel**([in] HOSTEX_TERM_MODEL *newVal*);

Parameters

pVal—The returned value, indicating the terminal model supported by HostExplorer.
newVal—The set value, indicating the terminal model supported by HostExplorer.

Basic Example

```
Dim OLE1 As Object
Dim ActiveSession As Object
Dim Parser As Object
Dim HOSTEX_TERM_MODEL_VAL As
HOSTEX_TERM_MODEL
Set OLE1 =
CreateObject("HostExplorer")
Set ActiveSession = OLE1.CurrentHost
Set Parser = ActiveSession.HEParserPtr
'get the value
HOSTEX_TERM_MODEL_VAL =
Parser.TerminalMode
'Set the value
Parser.TerminalMode =
HOSTEX_TERM_MODEL_TERM_IBM3151
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_TERM_MODEL HOSTEX_TERM_MODEL_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_TerminalMode

(&HOSTEX_TERM_MODEL_VAL);

// set value

pHEParser->put_TerminalMode

(HOSTEX_TERM_MODEL_TERM_IBM3151);

}
```

Related Topics

HOSTEX_TERM_MODEL Data Type

Property: IHEParser::TprintOutput 3270

This property returns or sets a value indicating where HostExplorer puts information received from a host TPRINT program. By default, this property is set to HOSTEX_TPRINT_OUTPUT_DEFAULT_WIN_PRINTER.

Basic Syntax

HOSTEX_TPRINT_OUTPUT = HEParser.**TPrintOutput**

HEParser.**TPrintOutput** = HOSTEX_TPRINT_OUTPUT

C++ Syntax

HRESULT IHEParser::**get_TPrintOutput**([out, retval] HOSTEX_TPRINT_OUTPUT * *pVal*);

HRESULT IHEParser::**put_TPrintOutput**([in] HOSTEX_TPRINT_OUTPUT *newVal*);

Parameters

pVal—The returned value, indicating where the print job is sent.

newVal—The set value, indicating where the print job is sent.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_TPRINT_OUTPUT_VAL As

HOSTEX_TPRINT_OUTPUT

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_TPRINT_OUTPUT_VAL =

Parser.TPrintOutput

'Set the value

Parser.TPrintOutput =

HOSTEX_TPRINT_OUTPUT_CLIPBOARD
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_TPRINT_OUTPUT HOSTEX_TPRINT_OUTPUT_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_TPrintOutput

(&HOSTEX_TPRINT_OUTPUT_VAL);

// set value

pHEParser->put_TPrintOutput

(HOSTEX_TPRINT_OUTPUT_CLIPBOARD);

}
```

Related Topics

HOSTEX_TPRINT_OUTPUT Data Type

Property: IHEParser::Transport 3270 5250 VT

This property returns or sets the callback pointer to the Transport interface.

Basic Syntax

Object = HEParser.**Transport**
HEParser.**Transport** = Object

C++ Syntax

HRESULT IHEParser::**get_Transport**([out, retval] IUnknown * * *pVal*);
HRESULT IHEParser::**put_Transport**([in] IUnknown * *newVal*);

Parameters

pVal—The returned callback pointer to the Transport interface.
newVal—The set callback pointer to the Transport interface.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim ITransport As Object

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

ITransport = Parser.Transport

'Set the value

Parser.Transport = ITransport
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

IUnknown * ITransport;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_Transport
```

```
(&ITransport);

// set value

pHEParser->put_Transport

(ITransport);

}
```

Property: IHEParser::TypeAheadTimeout 3270 5250

This property returns or sets the number (in milliseconds) that HostExplorer waits for a host response before canceling the attempt and clearing the type-ahead keyboard queue. By default, this option is set to 0, which means infinite timeout.

Basic Syntax

Long = HEParser.**TypeAheadTimeout**
HEParser.**TypeAheadTimeout** = Long

C++ Syntax

HRESULT IHEParser::**get_TypeAheadTimeout**([out, retval] long * *pVal*);
HRESULT IHEParser::**put_TypeAheadTimeout**([in] long *newVal*);

Parameters

pVal—The returned value, indicating the time delay.
newVal—The set value, indicating the time delay.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim TypeAhead As Long

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

TypeAhead = Parser.TypeAheadTimeout

'Set the value

Parser.PrinterTimeout = 5
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

long TypeAhead;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);
```

```
if (pHEParser())
{ // get value
pHEParser->get_TypeAheadTimeout
(&TypeAhead);
// set value
pHEParser->put_TypeAheadTimeout
(5);
}
```

Property: IHEParser::ValidateNumericFieldData 3270 5250

This property allows only numeric characters to be displayed in a numeric field. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**ValidateNumericFieldData**

HEParser.**ValidateNumericFieldData** = Boolean

C++ Syntax

HRESULT IHEParser::**get_ValidateNumericFieldData**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_ValidateNumericFieldData**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that only numeric characters are accepted in the input field. A returned value of VARIANT_FALSE indicates that any character can be accepted in the input field.

newVal—A set value of VARIANT_TRUE indicates that only numeric characters are accepted in the input field. A set value of VARIANT_FALSE indicates that any character can be accepted in the input field.

Basic Example

```
Dim OLE1 As Object
Dim ActiveSession As Object
Dim Parser As Object
Dim bValidator As Boolean
Set OLE1 =
CreateObject("HostExplorer")
Set ActiveSession = OLE1.CurrentHost
Set Parser = ActiveSession.HEParserPtr
'get the value
bValidator =
Parser.ValidateNumericFieldData
'Set the value
Parser.ValidateNumericFieldData = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL bValidator;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_ValidateNumericFieldData

(&bValidator);

// set value

pHEParser->put_ValidateNumericFieldData

(TRUE);

}
```

Property: IHEParser::VTMoveCursorOnMouseClick VT

This property lets you force HostExplorer to automatically move the cursor on a mouse-click. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**VTMoveCursorOnMouseClick**

HEParser.**VTMoveCursorOnMouseClick** = Boolean

C++ Syntax

HRESULT IHEParser::**get_VTMoveCursorOnMouseClick**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_VTMoveCursorOnMouseClick**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically moves the cursor on a mouse-click. A returned value of VARIANT_FALSE indicates that HostExplorer does not automatically move the cursor on a mouse-click.

newVal—A set value of VARIANT_TRUE indicates that HostExplorer automatically moves the cursor on a mouse-click. A set value of VARIANT_FALSE indicates that HostExplorer does not automatically move the cursor on a mouse-click.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim bMoveCurOnClick As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

bMoveCurOnClick =

Parser.VTMoveCursorOnMouseClick

'Set the value

Parser.VTMoveCursorOnMouseClick = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL OEM;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_VTMoveCursorOnMouseClick

(&bMoveCurOnClick);

// set value

pHEParser->put_VTMoveCursorOnMouseClick

(TRUE);

}
```

Property: IHEParser::VTNRC VT

This property returns or sets the NRC (National Replacement Character) set. By default, this property is set to 0.

Basic Syntax

Integer = HEParser.**VTNRC**
HEParser.**VTNRC** = Integer

C++ Syntax

HRESULT IHEParser::**get_VTNRC**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_VTNRC**([in] short *newVal*);

Parameters

pVal—The returned NRC set index.
newVal—The NRC set index that you specify.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iVTNRC As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iVTNRC = Parser.VTNRC

'Set the value

Parser.VTNRC = 1
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iVTNRC;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_VTNRC (&iVTNRC);

// set value

pHEParser->put_VTNRC (1);

}
```

Related Topics

Property: IHEParser::CharSet

TNVT NRC Language-Conversion Table

Property: IHEParser::VTScrollSpeed VT

This property returns or sets the scrolling speed (in pixels). By default, this property is set to 0.

Basic Syntax

Integer = HEParser.**VTScrollSpeed**
HEParser.**VTScrollSpeed** = Integer

C++ Syntax

HRESULT IHEParser::**get_VTScrollSpeed**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_VTScrollSpeed**([in] short *newVal*);

Parameters

pVal—The returned value, indicating the scrolling speed.
newVal—The set value, indicating the scrolling speed.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iScrollSpeed s As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iScrollSpeed = Parser.VTScrollSpeed

'Set the value

Parser.VTScrollSpeed = 3
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iScrollSpeed;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value
```

```
pHEParser->get_VTScrollSpeed

(&iTScrollSpeed);

// set value

pHEParser->put_VTScrollSpeed (3);

}
```

Property: IHEParser::VTUPSS VT

This property returns or sets the User Preferred Supplemental Character Set (UPSS). By default, this property is set to VTCS_ISO_8859_1.

Basic Syntax

Integer = HEParser.**VTUPSS**
HEParser.**VTUPSS** = Integer

C++ Syntax

HRESULT IHEParser::**get_VTUPSS**([out, retval] short * *pVal*);
HRESULT IHEParser::**put_VTUPSS**([in] short *newVal*);

Parameters

pVal—The returned UPSS index.
newVal—The set UPSS index.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim iVTUPSS As Integer

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

iVTUPSS = Parser.VTUPSS

'Set the value

Parser.VTUPSS = 234
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

short iVTUPSS;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_VTUPSS

(&iVTUPSS);

// set value

pHEParser->put_VTUPSS (234);

}
```

Related Topics

TNVT UPSS Language-Conversion Table

Property: IHEParser::WrapLines VT

This property indicates whether to automatically wrap lines that extend past the last column on the screen. The default value is VARIANT_FALSE.

Basic Syntax

Boolean = HEParser.**WrapLines**

HEParser.**WrapLines** = Boolean

C++ Syntax

HRESULT IHEParser::**get_WrapLines**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHEParser::**put_WrapLines**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically wraps lines that extend past the last column on the screen. A returned value of VARIANT_FALSE indicates that HostExplorer discards data that extends past the end of the screen.

newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically wraps lines that extend past the last column on the screen. A value of VARIANT_FALSE indicates that HostExplorer discards data that extends past the end of the screen.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim bWrapLines As Boolean

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

bWrapLines = Parser.WrapLines

'Set the value

Parser.WrapLines = True
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

BOOL bWrapLines;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_WrapLines

(&bWrapLines);

// set value

pHEParser->put_WrapLines (TRUE);

}
```

Property: IHEParser::XferErrorCode 3270

This property returns the error code from a file-transfer operation.

Basic Syntax

Long = HEParser.**XferErrorCode**

C++ Syntax

HRESULT IHEParser::**get_XferErrorCode**([out, retval] long * *pVal*);

Parameters

pVal—The returned error code.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim myErrorCode As Long

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

myErrorCode = Parser.XferErrorCode
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

long myErrorCode;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_XferErrorCode

(&myErrorCode);

}
```

Property: IHEParser::XferMode 3270

This property indicates whether to do an upload (send data to the host) or a download (retrieve data from the host). By default, this property is set to 0.

Basic Syntax

HOSTEX_TRANSFER = HEParser.**XferMode**
HEParser.**XferMode**(=HOSTEX_TRANSFER)

C++ Syntax

HRESULT IHEParser::**get_XferMode**([out, retval] HOSTEX_TRANSFER * *pVal*);
HRESULT IHEParser::**put_XferMode**([in] HOSTEX_TRANSFER newVal);

Parameters

pVal—The returned value, indicating whether to do an upload or download.
newVal—The set value, indicating whether to do an upload or download.

Basic Example

```
Dim OLE1 As Object

Dim ActiveSession As Object

Dim Parser As Object

Dim HOSTEX_TRANSFER_VAL As

HOSTEX_TRANSFER

Set OLE1 =

CreateObject("HostExplorer")

Set ActiveSession = OLE1.CurrentHost

Set Parser = ActiveSession.HEParserPtr

'get the value

HOSTEX_TRANSFER_VAL = Parser.XferMode

'Set the value

Parser.XferMode =

HOSTEX_TRANSFER_DOWNLOAD
```

C++ Example

```
HRESULT Hr;

IHEParser * pHEParser = NULL;

IHostExHost * pHost = NULL;

IHostExApplication * pApplication = NULL;

HOSTEX_TRANSFER HOSTEX_TRANSFER_VAL;



Hr = CoCreateInstance (CLSID_HostExApplication, NULL, CLSCTX_SERVER,
IID_IHostExApplication, (void **)&pApplication);

pApplication->get_CurrentHost(pHost);

pHost->get_HEParserPtr(&pHEParser);



if (pHEParser())

{ // get value

pHEParser->get_ XferMode

(&HOSTEX_TRANSFER_VAL);

// set value
```

```
pHEParser->put_XferMode
(HOSTEX_TRANSFER_DOWNLOAD);
}
```

Related Topics

HOSTEX_TRANSFER Data Type

Data Types of the Parser Objects

The Parser objects contain the following data types:

HEPARSER_FEATURE

HOSTEX_ATN_FORMAT

HOSTEX_CAPTURE_MODE

HOSTEX_CELL_DELIMITED

HOSTEX_CONNECT_BY

HOSTEX_CUT_MODE

HOSTEX_FIELD_ATTR_REPLACEMENT

HOSTEX_GRAPHICS_CELLSIZE

HOSTEX_GRAPHICS_CURSOR_TYPE

HOSTEX_GRAPHICS_MODEL

HOSTEX_INSERT_KEY_STYLE

HOSTEX_KEYBOARD_TYPE

HOSTEX_LINEMODE

HOSTEX_NEXT_FIELD_KEY

HOSTEX_PASTE_MODE

HOSTEX_SELECTION_MODE

HOSTEX_STATUS_LINE_MODE

HOSTEX_SWITCHSCREENTYPE

HOSTEX_TELNETECHO

HOSTEX_TERMINAL_ID

HOSTEX_TERM_MODEL

HOSTEX_TPRINT_OUTPUT

HOSTEX_TRANSFER

HOSTEX_TRANSFER_HOSTTYPE

HOSTEX_TRANSFER_INITIALACTION

HOSTEX_TRANSFER_RECORDFORMAT

HOSTEX_TRANSFER_TARGET

HOSTEX_TRANSFER_TYPE

HOSTOVERWRITE_BEHAVIOUR

PC_OVERWRITE_BEHAVIOUR

HOSTEX_GRAPHICS_CELLSIZE Data Type 3270

The HOSTEX_GRAPHICS_CELLSIZE data type indicates the cell size of a character in pixels. By default, this data type is set to HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC.

It has the following values:

HOSTEX_GRAPHICS_CELLSIZE_ AUTOMATIC	Indicates that HostExplorer does not correctly display the graphics for the automatic cell size. HostExplorer reports a Presentation Space size equal to the actual window size.
HOSTEX_GRAPHICS_CELLSIZE_ NINE_BY_TWELVE	Indicates that the cell size of the character is 9 x 12 pixels.
HOSTEX_GRAPHICS_CELLSIZE_ NINE_BY_SIXTEEN	Indicates that the cell size of the character is 9 x 16 pixels.
HOSTEX_GRAPHICS_CELLSIZE_ NINE_BY_TWENTY_ONE	Indicates that the cell size of the character is 9 x 21 pixels.
HOSTEX_GRAPHICS_CELLSIZE_ THIRTEEN_BY_TWENTY_TWO	Indicates that the cell size of the character is 13 x 22 pixels.
HOSTEX_GRAPHICS_CELLSIZE_ THIRTEEN_BY_TWENTY_NINE	Indicates that the cell size of the character is 13 x 29 pixels.

Related Topics

Property: IHESessionGraphics::PSCellSize

About the Session Object

The Session object contains the values of general session-related configuration settings for session items such as terminals, graphics, and security. It consists of the a number of interfaces:

Session Interface

The Session interface lets you modify the main configuration settings and lets you access other session-related interfaces.

Methods

The Session interface consists of the following methods:

Load

LoadQuickKeyFile

Save

Properties

The Session interface consists of the following properties:

AddOIAToCapture
AllowTN3270E
AlternateScreen
AreaCode
AttnFormat
AutoMacroName
AutoRunMacroDelayTime
CharacterSpacing
ClearPassword
ConnectBy
CountryCode
CountryID
Cursor
DDEServerName
DeviceName
DirectToModem
Display
Editing
EmuTraceFilename
EnableEmuTracing
EnableHLLAPITracing
FileTransfer
Fonts
FullScreenMode
GlobalSettingsPath
Graphics
HostName
IntegerName
Port
PrintExplorer
PrintScreen
ProfileName
PromptOnClose
QueryShutdown
SaveFile
SaveProfOnClose
Schemes
Security
ShowDialupDlg
Sound
SYSREQasIACIP
TelnetEcho
TelnetName
Terminal
TerminalType
Timeout
TraceFilename
TrackMenu
TranslationTable
TypeAhead
TypeAheadTimeout
UponDisconnect
UseDialProp
UserDirectory
UserName
VariableWidthFont

Keyboard
LongName
LUName
ModemID
Mouse
Notify
Password
PCPrint
VTCharset
VTDoHostWindowSize
VTInitiateTelnetNegotiation
VTLineMode
VTPrint
WinDDEEnabled
WindowTitle

Method: IHESession::Load 3270 5250 VT

This method loads the specified profiles.

Basic Syntax Session.**Load**

C++ Syntax HRESULT IHESession::**Load**();

Parameters This method has no parameters.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Session.ProfileName = "C:\\aix.hep"

Session.Load
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr;

bstr = SysAllocString(OLESTR("C:\\aix.hep"));

pSess->put_ProfileName(bstr);

pSess->Load();

SysFreeString(bstr);
```

Method: IHESession::LoadQuickKeyFile 3270 5250 VT

This method loads a new set of Quick-Keys (multi-functional shortcuts) for the session.

Note: Quick-keys are shared between sessions; loading a new set of Quick-Keys from one session will affect other sessions.

Basic Syntax

Session.**LoadQuickKeyFile**(*FileName* As String) As Integer

C++ Syntax

HRESULT IHESession::**LoadQuickKeyFile**([in] BSTR *FileName*, [out, retval] short * *pRc*);

Parameters

FileName—The name of the Quick-Key file that is to be loaded

pRc—The returned value, which points to the return code. A returned value of 1 indicates that the file was successfully loaded. A returned value of 0 indicates that the file was not successfully loaded.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Session.ProfileName = "C:\\aix.hep"

Session.Load

Session.LoadQuickKeyFile "C:\\Login.QKV"
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

bstr = SysAllocString(OLESTR("C:\\aix.hep"));

pSess->put_ProfileName(bstr);

SysFreeString(bstr);

pSess->Load();

bstr = SysAllocString(OLESTR("C:\\Login.QKV"));

pSess->LoadQuickKeyFile(bstr);

SysFreeString(bstr);
```

Method: IHESession::Save 3270 5250 VT

This method saves and updates the profile file with the current values in the associated Session object.

Basic Syntax

Session.**Save**

C++ Syntax

HRESULT IHESession::**Save**();

Parameters

This method has no parameters.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Session.ProfileName = "C:\\aix.hep"

Session.Load

Terminal.Connected = True

Session.HostName = "sunset"

Session.Save
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

bstr = SysAllocString(OLESTR("C:\\aix.hep"));

pSess->put_ProfileName(bstr);

SysFreeString(bstr);

pSess->Load();

VARIANT_BOOL vbVal = VARIANT_TRUE;

pSess->put_Connected(vbVal);

...

pSess->Save();
```

Property: IHESession::AddOIAToCapture 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer captures the OIA (Operator Information Area) in every screen.

Basic Syntax

Boolean = Session.**AddOIAToCapture**

Session.**AddOIAToCapture** = Boolean

C++ Syntax

HRESULT IHESession::**get_AddOIAToCapture**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_AddOIAToCapture**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer captures the OIA. A returned value of VARIANT_FALSE indicates that HostExplorer does not capture the OIA.

newVal—A value of VARIANT_TRUE indicates that HostExplorer captures the OIA. A value of VARIANT_FALSE indicates that HostExplorer does not capture the OIA.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.AddOIAToCapture

If (bVal = False) Then

‘Set value

Session.AddOIAToCapture = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal = pSess->get_AddOIAToCapture(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_AddOIAToCapture(bVal);

}
```

Property: IHESession::AllowTN3270E 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer is running the session in an enhanced mode (Extended Protocol). By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = Session.**AllowTN3270E**

Session.**AllowTN3270E** = Boolean

C++ Syntax

HRESULT IHESession::**get_AllowTN3270E**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_AllowTN3270E**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer is running the session in an enhanced mode. A returned value of VARIANT_FALSE indicates that HostExplorer is not running the session in an enhanced mode.

newVal—A value of VARIANT_TRUE indicates that HostExplorer is running the session in an enhanced mode. A value of VARIANT_FALSE indicates that HostExplorer is not running the session in an enhanced mode.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.AllowTN3270E

If (bVal = False) Then

‘Set value

Session.AllowTN3270E = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal = pSess->get_AllowTN3270E (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_AllowTN3270E (bVal);

}
```

Property: IHESession::AlternateScreen 3270

This property returns or sets a value indicating whether to change the window (screen) to the alternate size. By default, this option is VARIANT_FALSE.

Basic Syntax

Boolean = Session.**AlternateScreen**

Session.**AlternateScreen** = Boolean

C++ Syntax

HRESULT IHESession::**get_AlternateScreen**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_AlternateScreen**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the screen always opens in the alternate (larger) screen mode. A returned value of VARIANT_FALSE indicates that the default screen opens.

newVal—A value of VARIANT_TRUE indicates that the screen always opens in the alternate (larger) screen mode. A value of VARIANT_FALSE indicates that the default screen opens.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.AlternateScreen

If (bVal = False) Then

‘Set value

Session.AlternateScreen = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal = pSess->get_AlternateScreen(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_AlternateScreen (bVal);

}
```

Property: IHESession::AreaCode 3270 5250 VT

This property returns or sets a value specifying the area code when you use a modem connection. By default, this property is set to 0.

Basic Syntax

Long = Session.**AreaCode**
Session.**AreaCode** = Long

C++ Syntax

HRESULT IHESession::**get_AreaCode**([out, retval] long * *pVal*);
HRESULT IHESession::**put_AreaCode**([in] long *newVal*);

Parameters

pVal—The returned value, which indicates the area code.
newVal—The set value, which indicates the area code.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Long

‘Get value

lVal = Session.AreaCode

If (lVal = 0) Then

‘Set value

Session.AreaCode = 514

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_AreaCode (&lVal);

if (lVal!=514)

{

lVal=514;

pSess->put_AreaCode (lVal);

}
```

Property: IHESession::AttnFormat 3270 5250 VT

This property returns or sets a value that enables compatibility with other servers or gateways.

Basic Syntax

HOSTEX_ATN_FORMAT = Session.**AttnFormat**
Session.**AttnFormat** = HOSTEX_ATN_FORMAT

C++ Syntax

HRESULT IHESession::**get_AttnFormat**([out, retval] HOSTEX_ATN_FORMAT * *pVal*);
HRESULT IHESession::**put_AttnFormat**([in] HOSTEX_ATN_FORMAT *newVal*);

Parameters

pVal—The returned value, which enables compatibility with other application formats.
newVal—The set value, which enables compatibility with other application formats.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim AttnFormat As HOSTEX_ATN_FORMAT

AttnFormat = Session.AttnFormat

If (AttnFormat=HOSTEX_ATN_FORMAT_IBM) Then

Session.AttnFormat =

HOSTEX_ATN_FORMAT_ATTACHMATE

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

HOSTEX_ATN_FORMAT AttnFormat;

pSess->get_AttnFormat(&AttnFormat);

If (AttnFormat == HOSTEX_ATN_FORMAT_IBM)

{

AttnFormat =

HOSTEX_ATN_FORMAT_ATTACHMATE;

put_AttnFormat(AttnFormat);

}
```

Related Topics

HOSTEX_ATN_FORMAT Data Type

Property: IHESession::AutoMacroName 3270 5250 VT

This property returns or sets a string specifying the full path name of a macro. HostExplorer saves the full path name of the macro and launches it automatically each time you launch a new session.

Basic Syntax

String = Session.**AutoMacroName**
Session.**AutoMacroName** = String

C++ Syntax

HRESULT IHESession::**get_AutoMacroName**([out, retval] BSTR **pVal*);
HRESULT IHESession::**put_AutoMacroName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, which indicates the full pathname of the macro.
newVal—The set string, which indicates the full pathname of the macro.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get string

strVal = Session.AutoMacroName

If (Len(strVal) = 0) Then

‘Set string

Session.AutoMacroName = "C:\\a.ebs"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_AutoMacroName(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\\a.ebs"));

pSess->put_AutoMacroName(bstr);

SysFreeString(bstr);

}
```

Property: IHESession::AutoRunMacroDelayTime 3270 5250 VT

This property returns or sets a value (in seconds) indicating how long you must wait (after you launch a session) for a specified macro to execute. By default, the delay is set to zero.

Basic Syntax

Long = Session.**AutoRunMacroDelayTime**
Session.**AutoRunMacroDelayTime** = Long

C++ Syntax

HRESULT IHESession::**get_AutoRunMacroDelayTime**([out, retval] long * *pVal*);
HRESULT IHESession::**put_AutoRunMacroDelayTime**([in] long *newVal*);

Parameters

pVal—The returned value, which indicates how long you must wait for a macro to execute.
newVal—The set value, which indicates how long you must wait for a macro to execute.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Long

‘Get value

lVal = Session.AutoRunMacroDelayTime

If (lVal = 0) Then

‘Set value

Session.AutoRunMacroDelayTime = 10

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_AutoRunMacroDelayTime (&lVal);

if (lVal==0)

{

lVal=10;

pSess->

put_AutoRunMacroDelayTime (lVal);

}
```

Property: IHESession::CharacterSpacing 3270 5250 VT

This property returns or sets a value that lets you change the character spacing for variable-width fonts

Basic Syntax

Integer = Session.**CharacterSpacing**
Session.**CharacterSpacing** = Integer

C++ Syntax

HRESULT IHESession::**get_CharacterSpacing**([out, retval short * *pVal*);
HRESULT IHESession::**put_CharacterSpacing**([in] short *newVal*);

Parameters

pVal—The returned value, which changes the character spacing.
newVal—The set value, which changes the character spacing.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim iVal As Integer

‘Get value

iVal = Session.CharacterSpacing

If (iVal = 0) Then

‘Set value

Session.CharacterSpacing = 2

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

short sVal;

pSess->get_CharacterSpacing(&sVal);

if (sVal==0)

{

sVal=5;

pSess->put_CharacterSpacing(sVal);

}
```

Property: IHESession::ClearPassword 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer saves the session password in encrypted format. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = Session.**ClearPassword**

Session.**ClearPassword** = Boolean

C++ Syntax

HRESULT IHESession::**get_ClearPassword**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_ClearPassword**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer saves the password in non-encrypted format. A returned value of VARIANT_FALSE indicates that HostExplorer saves the password in encrypted format.

newVal—A value of VARIANT_TRUE indicates that HostExplorer saves the password in non-encrypted format. A value of VARIANT_FALSE indicates that HostExplorer saves the password in encrypted format.

Basic Example-

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.ClearPassword

If (bVal = False) Then

‘Set value

Session.ClearPassword = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal = pSess->get_ClearPassword(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_ClearPassword(bVal);

}
```

Property: IHESession::ConnectBy 3270 5250 VT

This property returns or sets a value indicating the method of connection (connection protocol) between the client machine and the host. By default, the property is set to HOSTEX_CONNECT_BY_TELNET.

Basic Syntax

HOSTEX_CONNECT_BY = Session.**ConnectBy**

Session.**ConnectBy** = HOSTEX_CONNECT_BY

C++ Syntax

HRESULT IHESession::**get_ConnectBy**([out, retval] HOSTEX_CONNECT_BY * *pVal*);

HRESULT IHESession::**put_ConnectBy**([in] HOSTEX_CONNECT_BY *newVal*);

Parameters

pVal—The returned value, indicating the connection.

newVal—The set value, indicating the connection.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim ConnBy As HOSTEX_CONNECT_BY

ConnBy = Session.ConnectBy

If (ConnBy <> HOSTEX_CONNECT_BY_MSSNA) Then

Sess.ConnectBy =

HOSTEX_CONNECT_BY_MSSNA

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

HOSTEX_CONNECT_BY ConnBy;

pSess->get_ConnectBy(&ConnBy);

if (ConnBy != HOSTEX_CONNECT_BY_MSSNA)

{

ConnBy = HOSTEX_CONNECT_BY_MSSNA;

pSess->put_ConnectBy(ConnBy);

}
```

Related Topics

HOSTEX_CONNECT_BY data type

Property: IHESession::CountryCode 3270 5250 VT

This property returns or sets a value specifying the country code when you use a modem connection.

Basic Syntax

Long = Session.**CountryCode**
Session.**CountryCode** = Long

C++ Syntax

HRESULT IHESession::**get_CountryCode**([out, retval] long * *pVal*);
HRESULT IHESession::**put_CountryCode**([in] long *newVal*);

Parameters

pVal—The returned value, specifying the country code.
newVal—The set value, specifying the country code.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Integer

‘Get value

lVal = Session.CountryCode

If (lVal = 0) Then

‘Set value

Session.CountryCode = 9133

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_CountryCode (&lVal);

if (lVal ==0)

{

lVal =9133;

pSess->put_CountryCode (lVal);

}
```

Property: IHESession::CountryID 3270 5250 VT

This property returns or sets a value specifying the country ID when you use a modem connection.

Basic Syntax

Long = Session.**CountryID**
Session.**CountryID** = Long

C++ Syntax

HRESULT IHESession::**get_CountryID**([out, retval] long * *pVal*);
HRESULT IHESession::**put_CountryID**([in] long *newVal*);

Parameters

pVal—The returned value specifying the country ID.
newVal—The set value specifying the country ID.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Integer

‘Get value

lVal = Session.CountryID

If (lVal = 0) Then

‘Set value

Session.CountryID = 47

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_CountryID (&lVal);

if (lVal ==0)

{

lVal =47;

pSess->put_CountryID (lVal);

}
```

Property: IHESession::Cursor 3270 5250 VT

This property returns a pointer to the Cursor object to retrieve general cursor-related options.

Basic Syntax HESessionCursor = Session.**Cursor**

C++ Syntax HRESULT IHESession::**get_Cursor**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionCursor interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessionCursor As HESessionCursor

Set SessionCursor = Session.Cursor

SessionCursor.VTMoveCursorOnMouseClick = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionCursor * pCursor;

pSess->get_Cursor(&pCursor);

VARIANT_BOOL bVal = VARIANT_TRUE;

pCursor->

put_VTMoveCursorOnMouseClick (bVal);
```

Property: IHESession::DDEServerName 3270 5250 VT

This property returns or sets a string specifying the DDE (Dynamic Data Exchange) server name. By default, this property is set to HOSTEX.

Basic Syntax String = Session.**DDEServerName**
Session.**DDEServerName** = String

C++ Syntax HRESULT IHESession::**get_DDEServerName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_DDEServerName**([in] BSTR *newVal*);

Parameters *pVal*—The returned string, specifying the DDE server name.
newVal—The set string, specifying the DDE server name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get value

strVal = Session.DDEServerName

If (Len(strVal) = 0) Then

‘Set Value

Session.DDEServerName = "HOSTEX"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr;

pSess->get_DDEServerName(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("HOSTEX"));

pSess->put_DDEServerName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::DeviceName 5250

This property returns or sets a value indicating whether you want to connect to a default or specific device name.

Basic Syntax

String = Session.**DeviceName**
Session.**DeviceName** = String

C++ Syntax

HRESULT IHESession::**get_DeviceName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_DeviceName**([in] BSTR *newVal*);

Parameters

pVal—The returned value, which indicates whether to connect to a default or specific device name.
newVal—The set value, which indicates whether you want to connect to a default or specific device name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get value

strVal = Session.DeviceName

If (Len(strVal) = 0) Then

‘Set Value

Session.DeviceName = "TTS"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_DeviceName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("TTS"));

pSess->put_DeviceName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::DirectToModem VT

This property returns or sets a value indicating whether HostExplorer sets the DirectToModem flag. When set, this flag causes the emulator to connect to the modem immediately and to bypass the dialing stage. This lets you use a modem link as if it were a COM port.

Basic Syntax

Boolean = Session.**DirectToModem**
Session.**DirectToModem** = Boolean

C++ Syntax

HRESULT IHESession::**get_DirectToModem**([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHESession::**put_DirectToModem**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the DirectToModem flag is set. A returned value of VARIANT_FALSE indicates that the DirectToModem flag is not set.
newVal—A value of VARIANT_TRUE indicates that the DirectToModem flag is set. A value of VARIANT_FALSE indicates that the DirectToModem flag is not set.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.DirectToModem

If (bVal = False) Then

‘Set value

Session.DirectToModem = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal = pSess->get_DirectToModem(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_DirectToModem(bVal);

}
```

Property: IHESession::Display 3270 5250 VT

This property returns a pointer to the SessionDisplay interface to retrieve general display-related options.

Basic Syntax HESessionDisplay = Session.**Display**

C++ Syntax HRESULT IHESession::**get_Display**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionDisplay interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessionDisplay As HESessionDisplay

Set SessionDisplay = Session.Display

SessionDisplay.AlwaysAutoskip = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionDisplay* pSessDisplay;

pSess->get_Display(&pSessDisplay);

VARIANT_BOOL bVal = VARIANT_TRUE;

pSessDisplay->put_AlwaysAutoskip (bVal);
```

Property: IHESession::Editing 3270 5250 VT

This property returns a pointer to the SessionEditing interface to retrieve general editing-related options.

Basic Syntax

```
HESessionEditing = Session.Editing
```

C++ Syntax

```
HRESULT IHESession::get_Editing([out, retval] IUnknown * * pVal);
```

Parameters

pVal—The returned pointer to the IHESessionEditing interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessionEditing As HESessionEditing

Set SessionEditing = Session.Editing

SessionEditing.AutoCopy = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionEditing * pSessEdit;

pSess->get_Editing(&pSessEdit);

VARIANT_BOOL bVal = VARIANT_TRUE;

pSessEdit->put_AutoCopy (bVal);
```

Property: IHESession::EmuTraceFilename 3270 5250 VT

This property returns or sets a string specifying the name of the tracing file. By default, this file is located in C:\EHLLAPI.TRC.

Basic Syntax

String = Session.**EmuTraceFilename**

Session.**EmuTraceFilename** = String

C++ Syntax

HRESULT IHESession::**get_EmuTraceFilename**([out, retval] BSTR * *pVal*);

HRESULT IHESession::**put_EmuTraceFilename**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the name of the tracing file, including the full pathname.

newVal—The set string, specifying the name of the tracing file, including the full pathname.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get value

strVal = Session.EmuTraceFilename

If (Len(strVal) = 0) Then

‘Set value

Session.EmuTraceFilename =

“HETrace.trc"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_EmuTraceFilename (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("HETrace.trc"));

pSess->put_EmuTraceFilename (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::EnableEmuTracing 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer is running the emulator-tracing function. By default, this option is set to VARIANT_FALSE.

Basic Syntax

Boolean = Session.**EnableEmuTracing**

Session.**EnableEmuTracing** = Boolean

C++ Syntax

HRESULT IHESession::**get_EnableEmuTracing**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_EnableEmuTracing**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer is running the emulator-tracing function. A returned value of VARIANT_FALSE indicates that HostExplorer is not running the emulator-tracing function.

newVal—A value of VARIANT_TRUE indicates that HostExplorer is running the emulator-tracing function. A value of VARIANT_FALSE indicates that HostExplorer is not running the emulator-tracing function.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.EnableEmuTracing

If (bVal = False) Then

‘Set value

Session.EnableEmuTracing = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal = pSess->get_EnableEmuTracing (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_EnableEmuTracing(bVal);

}
```

Property: IHESession::EnableHLLAPITracing 3270 5250 VT

This property returns or sets a value that specifies whether you want to enable tracing (the process of writing the host information to a file).

Basic Syntax

Boolean = Session.**EnableHLLAPITracing**

Session.**EnableHLLAPITracing** = Boolean

C++ Syntax

HRESULT IHESession::**get_EnableHLLAPITracing**([out, retval] VARIANT_BOOL *
pVal);

HRESULT IHESession::**put_EnableHLLAPITracing**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE enables tracing. A returned value of VARIANT_FALSE disables tracing.

newVal—A value of VARIANT_TRUE enables tracing. A value of VARIANT_FALSE disables tracing.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.EnableHLLAPITracing

If (bVal = False) Then

‘Set value

Session.EnableHLLAPITracing = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_EnableHLLAPITracing (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_EnableHLLAPITracing(bVal);

}
```

Property: IHESession::FileTransfer 3270 5250 VT

This property returns a pointer to the SessionFileTransfer interface to retrieve general file-transfer options.

Basic Syntax HESessionFileTransfer = Session.**FileTransfer**

C++ Syntax HRESULT IHESession::**get_FileTransfer**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionFileTransfer interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessFileXfer As HESessionFileTransfer

Set SessFileXfer = Session.FileTransfer

SessionFileXfer.Append = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionFileTransfer * pXfer;

pSess->get_FileTransfer (&pXfer);

VARIANT_BOOL bVal = VARIANT_TRUE;

pXfer->put_Append (bVal);
```

Property: IHESession::Fonts 3270 5250 VT

This property returns a pointer to the SessionFonts interface to retrieve general font-related options (such as Font Name, Font Size, and Font Style).

Basic Syntax HESessionFonts = Session.**Fonts**

C++ Syntax HRESULT IHESession::**get_Fonts**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionFonts interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessFonts As HESessionFonts

Set SessFonts = Session.Fonts

SessFonts.SaveFontOnExit = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionFonts * pFonts;

pSess->get_Fonts(&pFonts);

VARIANT_BOOL bVal = VARIANT_TRUE;

pFonts->put_SaveFontOnExit (bVal);
```

Property: IHESession::FullScreenMode 3270 5250 VT

This property returns or sets a value indicating whether the session will run in full-screen or regular-screen display mode.

Basic Syntax

Boolean = Session.**FullScreenMode**

Session.**FullScreenMode** = Boolean

C++ Syntax

HRESULT IHESession::**get_FullScreenMode**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_FullScreenMode**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the session will run in full-screen display mode. A returned value of VARIANT_FALSE indicates that the session will run in regular-screen display mode.

newVal—A set value of VARIANT_TRUE indicates that the session will run in full-screen display mode. A set value of VARIANT_FALSE indicates that the session will run in regular-screen display mode.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.FullScreenMode

If (bVal = False) Then

‘Set value

Session.FullScreenMode = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_FullScreenMode (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_FullScreenMode (bVal);

}
```

Property: IHESession::GlobalSettingsPath 3270 5250 VT

This property returns or sets a string specifying the full path and name of the global settings file. By default, this property is set to HostEx.ini.

Basic Syntax

String = Session.**GlobalSettingsPath**
Session.**GlobalSettingsPath** = String

C++ Syntax

HRESULT IHESession::**get_GlobalSettingsPath**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_GlobalSettingsPath**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the full path and name of the global settings file.
newVal—The set string, specifying the full path and name of the global settings file.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get path

strVal = Session.GlobalSettingsPath

If (Len(strVal) = 0) Then

‘Set path

Session.GlobalSettingsPath = "C:\\HostEx.ini"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_GlobalSettingsPath (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\\HostEx.ini

"));

pSess->put_GlobalSettingsPath (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::Graphics 3270 5250 VT

This property returns a pointer to the SessionGraphics interface to retrieve general graphic-related options.

Basic Syntax HESessionGraphics = Session.**Graphics**

C++ Syntax HRESULT IHESession::**get_Graphics**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionGraphics interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessGraphics As HESessionGraphics

Set SessGraphics = Session.Graphics

SessGraphics.APL = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionGraphics * pGraphics;

pSess->get_Graphics (&pFonts);

VARIANT_BOOL bVal = VARIANT_TRUE;

pGraphics->put_Graphics(bVal);
```

Property: IHESession::HostName 3270 5250 VT

This property returns or sets a string that specifies the name of the host to which you are trying to connect.

Basic Syntax

String = Session.**HostName**
Session.**HostName** = String

C++ Syntax

HRESULT IHESession::**get_HostName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_HostName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, indicating the host name.
newVal—The set string, indicating the host name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get string

strVal = Session.HostName

If (Len(strVal) = 0) Then

‘Set string

Session.HostName = "aix"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_HostName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("aix"));

pSess->put_HostName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::Keyboard 3270 5250 VT

This property returns a pointer to the SessionKeyboard interface to retrieve general keyboard-related options.

Basic Syntax SessionKeyboard = Session.**Keyboard**

C++ Syntax HRESULT IHESession::**get_Keyboard**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionKeyboard interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessKeyboard As HESessionKeyboard

Set SessKeyboard = Session.Keyboard

SessKeyboard.AllowDiac = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionKeyboard * pKeyboard;

pSess->get_Keyboard (&pKeyboard);

VARIANT_BOOL bVal = VARIANT_TRUE;

pKeyboard->put_AllowDiac (bVal);
```

Property: IHESession::LongName 3270 5250 VT

This property returns or sets a string specifying the session Long name that appears in the OIA (Operator Information Area). The Long name can contain up to eight characters.

Basic Syntax

String = Session.**LongName**
Session.**LongName** = String

C++ Syntax

HRESULT IHESession::**get_LongName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_LongName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the current session Long name.
newVal—The set string, specifying the current session Long name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get value

strVal = Session.LongName

If (Len(strVal) = 0) Then

‘Set value

Session.LongName = "Sess-1"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_LongName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("Sess-1"));

pSess->put_LongName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::LUName 3270

This property returns or sets a string specifying the LU (Logical Units) name that is used when you connect to a host that supports RFC1646 or RFC2205 (TN3270). An LU contains the necessary information needed to connect to an SNA network.

Basic Syntax

String = Session.**LUName**
Session.**LUName** = String

C++ Syntax

HRESULT IHESession::**get_LUName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_LUName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the LU name.
newVal—The set string, specifying the LU name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

‘Get string

strVal = Session.LUName

If (Len(strVal) = 0) Then

‘Set string

Session.LUName = "XYZ"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_LUName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("XYZ"));

pSess->put_LUName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::ModemID 3270 5250 VT

This property returns or sets a value specifying the modem ID when you use a modem connection.

Basic Syntax

```
Long = Session.ModemID
Session.ModemID = Long
```

C++ Syntax

```
HRESULT IHESession::get_ModemID([out, retval] long * pVal);
HRESULT IHESession::put_ModemID([in] long newVal);
```

Parameters

pVal—The returned value, specifying the modem ID.
newVal—The set value, specifying the modem ID.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Long

lVal = Session.ModemID

If (lVal = 6734) Then

‘Add code

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_ModemID (&lVal);

if (lVal ==0)

{

lVal =6734;

pSess->put_ModemID (lVal);

}
```

Property: IHESession::Mouse 3270 5250 VT

This property returns a pointer to the SessionMouse interface to retrieve general mouse-related options.

Basic Syntax HESessionMouse = Session.**Mouse**

C++ Syntax HRESULT IHESession::**get_Mouse**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionMouse interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessMouse As HESessionMouse

Set SessMouse = Session.Mouse

SessMouse.BlockSelect = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionMouse * pMouse;

pSess->get_Mouse(&pMouse);

VARIANT_BOOL bVal = VARIANT_TRUE;

pMouse->put_BlockSelect (bVal);
```

Property: IHESession::Notify 3270 5250 VT

This property returns or sets a value indicating whether the Notify (Update Alarm) option is on or off. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = Session.**Notify**

Session.**Notify** = Boolean

C++ Syntax

HRESULT IHESession::**get_Notify**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_Notify**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Notify option is on. A returned value of VARIANT_FALSE indicates that the Notify option is off.

newVal—A value of VARIANT_TRUE indicates that the Notify option is on. A value of VARIANT_FALSE indicates that the Notify option is off.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

‘Get value

bVal = Session.Notify

If (bVal = False) Then

‘Set value

Session.Notify = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_Notify (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_Notify (bVal);

}
```

Property: IHESession::Password 5250

This property returns or sets a string that specifies the password used to connect to the host.

Basic Syntax

String = Session.**Password**
Session.**Password** = String

C++ Syntax

HRESULT IHESession::**get_Password**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_Password**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the password.
newVal—The set string, specifying the password.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

strVal = Session.Password

If (Len(strVal) = 0) Then

Session.Password = "as14rty"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_Password (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("as14rty "));

pSess->put_Password (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::PCPrint 3270 5250 VT

This property returns a pointer to the SessionPCPrint interface to retrieve general output-related options.

Basic Syntax
HESessionPCPrint = Session.**PCPrint**

C++ Syntax
HRESULT IHESession::**get_PCPrint**([out, retval] IUnknown * * *pVal*);

Parameters
pVal—The returned pointer to the IHESessionPCPrint interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessPCPrint As HESessionPCPrint

Set SessPCPrint = Session.PCPrint

SessPCPrint.TprintMode = HOSTEX_TPRINT_OUTPUT_LPT1
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESession PCPrint * pPCPrint;

pSess->get_PCPrint (&pPCPrint);

pPCPrint->put_TprintMode (HOSTEX_TPRINT_OUTPUT_LPT1);
```

Property: IHESession::Port 3270 5250 VT

This property returns or sets a value that specifies the Internet port to which you are connected. You can select a number between 1 and 65534. By default, this property is set to 23.

Basic Syntax
Integer = Session.**Port**
Session.**Port** = Integer

C++ Syntax
HRESULT IHESession::**get_Port**([out, retval] short * *pVal*);
HRESULT IHESession::**put_Port**([in] short *newVal*);

Parameters
pVal—The returned value, specifying the Internet port.
newVal—The set value, specifying the Internet port.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim iVal As Integer

‘Get value

iVal = Session.Port

If (iVal <> 23) Then

‘Set value

Session.Port = 23

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

short sVal;

pSess->get_Port (&sVal);

if (sVal==0)

{

sVal=23;

pSess->put_Port (sVal);

}
```

Property: IHESession::PrintExplorer 3270 5250 VT

This property returns a pointer to the SessionPrintExplorer interface to retrieve options related to launching a PrintExplorer session.

Basic Syntax HESessionPrintExplorer = Session.**PrintExplorer**

C++ Syntax HRESULT IHESession::**get_PrintExplorer**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionPrintExplorer interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessPrintEx As HESessionPrintExplorer

Set SessPrintEx = Session.PrintExplorer

SessPrintEx.PEHostName = "sunset.cs.ca"
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionPrintExplorer * pPExplorer;

pSess->get_PExplorer(&pPExplorer);

BSTR bstr = SysAllocString(OLESTR("sunset.cs.ca "));
pPExplorer->put_PEHostName (bstr);
```

Property: IHESession::PrintScreen 3270 5250 VT

This property returns a pointer to the SessionPrintScreen object to retrieve general options related to printing the screen.

Basic Syntax HESessionPrintScreen = Session.**PrintScreen**

C++ Syntax HRESULT IHESession::**get_PrintScreen**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionPrintScreen interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessPrintScrn As HESessionPrintScreen

Set SessPrintScrn = Session.PrintScreen

SessPrintScrn.PrintOIA = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionPrintScreen * pPrintScrn;

pSess->get_PrintScrn(&pPrintScrn);

VARIANT_BOOL bVal = VARIANT_TRUE;

pPrintScrn->put_PrintOIA(bVal);
```

Property: IHESession::ProfileName 3270 5250 VT

This property returns or sets a string indicating the profile name (including the full pathname) for a session.

Basic Syntax

String = Session.**ProfileName**
Session.**ProfileName** = String

C++ Syntax

HRESULT IHESession::**get_ProfileName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_ProfileName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, which indicates the profile name.
newVal—The set string, which indicates the profile name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

'Get string

strVal = Session.ProfileName

If (Len(strVal) = 0) Then

'Set string

Session.ProfileName = "C:\\aix.hep"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_ProfileName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\\aix.hep "));

pSess->put_ProfileName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::PromptOnClose 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prompts you before closing a window session. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = Session.**PromptOnClose**

Session.**PromptOnClose** = Boolean

C++ Syntax

HRESULT IHESession::**get_PromptOnClose**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_PromptOnClose**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer prompts you before closing a window session. A returned value of VARIANT_FALSE indicates that HostExplorer does not prompt you before closing a window session.

newVal—A value of VARIANT_TRUE indicates that HostExplorer prompts you before closing a window session. A value of VARIANT_FALSE indicates that HostExplorer does not prompt you before closing a window session.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.PromptOnClose

If (bVal = False) Then

'Set value

Session.PromptOnClose = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_PromptOnClose(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_PromptOnClose(bVal);

}
```

Property: IHESession::QueryShutdown 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer shuts down automatically if it is still active when Windows shuts down. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = Session.**QueryShutdown**

Session.**QueryShutdown** = Boolean

C++ Syntax

HRESULT IHESession::**get_QueryShutdown**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_QueryShutdown**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer forces Windows to postpone shutting down until HostExplorer fully shuts down. A returned value of VARIANT_FALSE indicates that HostExplorer shuts down automatically when Windows shuts down.

newVal—A value of VARIANT_TRUE indicates that HostExplorer forces Windows to postpone shutting down until HostExplorer fully shuts down. A value of VARIANT_FALSE indicates that HostExplorer shuts down automatically when Windows shuts down.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.QueryShutdown

If (bVal = False) Then

'Set value

Session.QueryShutdown = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_QueryShutdown (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_QueryShutdown (bVal);

}
```

Property: IHESession::SaveFile 3270 5250 VT

This property returns a pointer to the SessionSaveFile interface to retrieve the Save Filename option.

Basic Syntax
HESessionSaveFile = Session.**SaveFile**

C++ Syntax
HRESULT IHESession::**get_SaveFile**([out, retval] IUnknown * * *pVal*);

Parameters
pVal—The returned pointer to the IHESessionSaveFile interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessSave As HESessionSaveFile

Set SessSave = Session.SaveFile

SessSave.SaveConfirm = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionSaveFile * pSaveFile;

pSess->get_SaveFile(&pSaveFile);

VARIANT_BOOL bVal = VARIANT_TRUE;

pSaveFile->put_SaveConfirm (bVal);
```

Property: IHESession::SaveProfOnClose 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer automatically saves any setting changes for the current profile when you close a session. By default, this property is set to VARIANT_TRUE.

Basic Syntax
Boolean = Session.**SaveProfOnClose**
Session.**SaveProfOnClose** = Boolean

C++ Syntax
HRESULT IHESession::**get_SaveProfOnClose**([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHESession::**put_SaveProfOnClose**([in] VARIANT_BOOL *newVal*);

Parameters
pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically saves the setting changes for the current profile when you close the session. A returned value of VARIANT_FALSE indicates that HostExplorer does not save the setting changes for the current profile when you close the session.
newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically saves the setting changes for the current profile when you close the session. A value of VARIANT_FALSE indicates that HostExplorer does not save the setting changes for the current profile when you close the session.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.SaveProfOnClose

If (bVal = False) Then

'Set value

Session.SaveProfOnClose = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_SaveProfOnClose (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_SaveProfOnClose (bVal);

}
```

Property: IHESession::Schemes 3270 5250 VT

This property returns the pointer to the SessionSchemes interface. A scheme is a collection of settings.

Basic Syntax SessionSchemes = Session.**Schemes**

C++ Syntax HRESULT IHESession::**get_Schemes**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionSchemes interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessSc As HESessionSchemes

Set SessSc = Session.Schemes
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionSchemes * pSchemes;

pSess->get_Schemes (&pSchemes);
```

Property: IHESession::Security 3270 5250 VT

This property returns a pointer to the SessionSecurity interface to retrieve general security-related options.

Basic Syntax HESessionSecurity = Session.**Security**

C++ Syntax HRESULT IHESession::**get_Security**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionSecurity interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessSec As HESessionSecurity

Set SessSec = Session.Security
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionSecurity * pSessSec;

pSess->get_Security(&pSessSec);
```

Property: IHESession::IntegerName 3270 5250 VT

This property returns or sets a string specifying the session short name. The session short name is used primarily by HLLAPI applications. Changing the short name of a session takes effect immediately and does not require the emulator or session to be restarted.

Basic Syntax

String = Session.**IntegerName**
Session.**IntegerName** = String

C++ Syntax

HRESULT IHESession::**get_IntegerName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_IntegerName**([in] BSTR *newVal*);

Parameters

pVal—The returned letter of the alphabet, indicating the current HLLAPI short name.
newVal—The set letter of the alphabet, indicating the current HLLAPI short name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

'Get string

strVal = Session.IntegerName

If (Len(strVal) = 0) Then

'Set string

Session.IntegerName = "G"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_IntegerName(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("G"));

pSess->put_IntegerName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::ShowDialupDlg VT

This property returns or sets a value indicating whether HostExplorer displays the Always Show Connect Dialog flag, prompting you to verify modem properties. This property does not affect connections that are started from OLE Automation. This property is used only for returned or set values so that you can load or save profiles. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = Session.**ShowDialupDlg**

Session.**ShowDialupDlg** = Boolean

C++ Syntax

HRESULT IHESession::**get_ShowDialupDlg**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_ShowDialupDlg**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays the Always Show Connect Dialog flag. A returned value of VARIANT_FALSE indicates that HostExplorer does not display the Always Show Connect Dialog flag.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays the Always Show Connect Dialog flag. A value of VARIANT_FALSE indicates that HostExplorer does not display the Always Show Connect Dialog flag.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.ShowDialupDlg

If (bVal = False) Then

'Set value

Session.ShowDialupDlg = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_ShowDialupDlg (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_ShowDialupDlg (bVal);

}
```

Property: IHESession::Sound 3270 5250 VT

This property returns or sets a value indicating whether the Sound option for program sounds is on or off. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = Session.**Sound**

Session.**Sound** = Boolean

C++ Syntax

HRESULT IHESession::**get_Sound**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_Sound**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Sound option for program sounds is on. A returned value of VARIANT_FALSE indicates that the Sound option for program sounds is off.

newVal—A value of VARIANT_TRUE indicates that the Sound option for program sounds is on. A value of VARIANT_FALSE indicates that the Sound option for program sounds is off.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.Sound

If (bVal = False) Then

'Set value

Session.Sound = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_Sound (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_Sound (bVal);

}
```

Property: IHESession::SYSREQasIACIP 3270

This property returns or sets a value indicating whether the SYSREQ key as the Telnet IAC IP sequence is enabled or disabled. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = Session.**SYSREQasIACIP**

Session.**SYSREQasIACIP** = Boolean

C++ Syntax

HRESULT IHESession::**get_SYSREQasIACIP**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_SYSREQasIACIP**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the SYSREQ key as the Telnet IAC IP sequence is enabled. A returned value of VARIANT_FALSE indicates that the sequence is disabled.

newVal—A value of VARIANT_TRUE indicates that the SYSREQ key as the Telnet IAC IP sequence is enabled. A value of VARIANT_FALSE indicates that the sequence is disabled.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.SYSREQasIACIP

If (bVal = False) Then

'Set value

Session.SYSREQasIACIP = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_SYSREQasIACIP (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_SYSREQasIACIP (bVal);

}
```

Property: IHESession::TelnetEcho VT

This property returns or sets a value that indicates how HostExplorer responds to remote echo negotiation with a Telnet host.

Basic Syntax

HOSTEX_TELNETECHO = Session.**TelnetEcho**

Session.**TelnetEcho** = HOSTEX_TELNETECHO

C++ Syntax

HRESULT IHESession::**get_TelnetEcho**([out, retval] HOSTEX_TELNETECHO * *pVal*);

HRESULT IHESession::**put_TelnetEcho**([in] HOSTEX_TELNETECHO *newVal*);

Parameters

pVal—The returned value, which indicates how HostExplorer responds to remote echo negotiation with a Telnet host.

newVal—The set value, which indicates how HostExplorer responds to remote echo negotiation with a Telnet host.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim TEcho As HOSTEX_TELNETECHO

‘Get value

TEcho = Session.TelnetEcho

If (TEcho = HOSTEX_TELNETECHO_NO) Then

‘Set value

Session.TEcho = HOSTEX_TELNETECHO_YES

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

HOSTEX_TELNETECHO TEcho;

pSess->get_TelnetEcho(&TEcho);

if (TEcho!= HOSTEX_TELNETECHO_NO)

{

TEcho = HOSTEX_TELNETECHO_NO;

pSess->put_TelnetEcho (TEcho);

}
```

Related Topics

HOSTEX_TELNETECHO Data Type

Property: IHESession::TelnetName VT

This property returns or sets a string that specifies a name to override the name used during Telnet negotiation with the host.

Basic Syntax

String = Session.**TelnetName**
Session.**TelnetName** = String

C++ Syntax

HRESULT IHESession::**get_TelnetName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_TelnetName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying a Telnet name.
newVal—The set string, specifying a Telnet name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

'Get string

strVal = Session.TelnetName

If (Len(strVal) = 0) Then

'Set string

Session.TelnetName = "sunset"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_TelnetName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("sunset"));

pSess->put_TelnetName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::Terminal 3270 5250 VT

This property returns a pointer to the SessionTerminal interface and its associated variable values in HostExplorer.

Basic Syntax
HESessionTerminal = Session.**Terminal**

C++ Syntax
HRESULT IHESession::**get_Terminal**([out, retval] IUnknown * * *pVal*);

Parameters
pVal—The returned pointer to the IHESessionTerminal interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

SessTerm.ReplyOEM = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESession Terminal * pTerm;

pSess->get_Terminal (&pTerm);

VARIANT_BOOL bVal = VARIANT_TRUE;

pTerm->put_ReplyOEM (bVal);
```

Property: IHESession::TerminalType 3270 5250 VT

This property returns or sets a value specifying the terminal type for the session, such as TN3270, TN5250, TNVT, VT-52, or VT-100.

Basic Syntax
Long = Session.**TerminalType**
Session.**TerminalType** = Long

C++ Syntax
HRESULT IHESession::**get_TerminalType**([out, retval] long * *pVal*);
HRESULT IHESession::**put_TerminalType**([in] long *newVal*);

Parameters
pVal—The following returned values specify the terminal type for the session:

1—TN3270 terminals

2—TNVT terminals

3—TN5250 terminals
newVal—The set value, which indicates the terminal type for the session.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Long

'Get value

lVal = Session.TerminalType

If (lVal <> 2) Then

'Set value

Session.TerminalType = 2

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_TerminalType (&lVal);

if (lVal != 2)

{

lVal =2;

pSess->put_TerminalType (lVal);

}
```

Property: IHESession::Timeout 3270 5250 VT

This property returns or sets a value specifying the number of seconds required for HostExplorer to establish a connection before canceling the operation. By default, this property is set to 30.

Basic Syntax

```
Long = Session.Timeout
Session.Timeout = Long
```

C++ Syntax

```
HRESULT IHESession::get_Timeout([out, retval] long * pVal);
HRESULT IHESession::put_Timeout([in] long newVal);
```

Parameters

pVal—The returned value, specifying the number of seconds required for HostExplorer to establish a connection.

newVal—The set value, specifying the number of seconds required for HostExplorer to establish a connection.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Long

'Get value

lVal = Session.Timeout

If (lVal = 0) Then

'Set value

Session.Timeout = 100

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_Timeout (&lVal);

if (lVal == 0)

{

lVal =2;

pSess->put_Timeout (100);

}
```

Property: IHESession::TraceFilename 3270 5250 VT

This property returns or sets a string specifying the name of the file that contains the information that is sent to and received from the host.

Basic Syntax

String = Session.**TraceFilename**
Session.**TraceFilename** = String

C++ Syntax

HRESULT IHESession::**get_TraceFilename**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_TraceFilename**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the file name.
newVal—The set string, specifying the file name.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

'Get string

strVal = Session.TraceFilename

If (Len(strVal) = 0) Then

'Set string

Session.TraceFilename =

"C:\\trace.trc"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_TraceFilename (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\\trace.trc"));

pSess->put_TraceFilename (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::TrackMenu 3270 5250 VT

This property returns a pointer to the SessionTrackMenu interface to retrieve general options related to the Track menu.

Basic Syntax HESessionTrackMenu = Session.**TrackMenu**

C++ Syntax HRESULT IHESession::**get_TrackMenu**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionTrackMenu interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessTrack As HESessionTrackMenu

Set SessTrack = Session.TrackMenu
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionTrackMenu * pTrackMenu;

pSess->get_TrackMenu(&pTrackMenu);
```

Property: IHESession::TranslationTable 3270 5250 VT

This property returns a pointer to the SessionTranslationTable interface to retrieve TranslationTable options related to the language used and whether the file being transferred is text or binary.

Basic Syntax HESessionTranslationTable = Session.**TranslationTable**

C++ Syntax HRESULT IHESession::**get_TranslationTable**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionTranslationTable interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessTrans As HESessionTranslationTable

Set SessTrans = Session.TranslationTable

SessTrans.UsePrivateTranslate = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionTranslationTable * pTrans;

pSess->

get_TranslationTable (&pTrans);

VARIANT_BOOL bVal = VARIANT_TRUE;

pTrans->put_UsePrivateTranslate(bVal);
```

Property: IHESession::TypeAhead 3270 5250

This property returns or sets a value indicating whether HostExplorer buffers typed characters, even when the keyboard is locked. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = Session.**TypeAhead**
Session.**TypeAhead** = Boolean

C++ Syntax

HRESULT IHESession::**get_TypeAhead**([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHESession::**put_TypeAhead**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer buffers typed characters. A returned value of VARIANT_FALSE indicates that HostExplorer is not buffering typed characters.
newVal—A value of VARIANT_TRUE indicates that HostExplorer buffers typed characters. A value of VARIANT_FALSE indicates that HostExplorer is not buffering typed characters.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.TypeAhead

If (bVal = False) Then

'Set value

Session.TypeAhead = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_TypeAhead (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_TypeAhead (bVal);

}
```

Property: IHESession::TypeAheadTimeout 3270 5250

This property returns or sets the amount of time (in milliseconds) that HostExplorer waits for a host response before canceling an attempt and clearing the Type Ahead keyboard queue. By default, this property is set to 0, which means infinite timeout.

Basic Syntax

Long = Session.**TypeAheadTimeout**
Session.**TypeAheadTimeout** = Long

C++ Syntax

HRESULT IHESession::**get_TypeAheadTimeout**([out, retval] long * *pVal*);
HRESULT IHESession::**put_TypeAheadTimeout**([in] long *newVal*);

Parameters

pVal—The returned value, which indicates the time delay.
newVal—The set value, which indicates the time delay.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim lVal As Long

'Get value

lVal = Session.TypeAheadTimeout

If (lVal = 0) Then

'Set value

Session.TypeAheadTimeout = 100

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

long lVal;

pSess->get_TypeAheadTimeout (&lVal);

if (lVal == 0)

{

lVal =100;

pSess->put_TypeAheadTimeout (lVal);

}
```

Property: IHESession::UponDisconnect 3270 5250 VT

This property returns or sets a value specifying the action that HostExplorer performs if the host disconnects from the current session.

Basic Syntax

Integer = Session.**UponDisconnect**
Session.**UponDisconnect** = Integer

C++ Syntax

HRESULT IHESession::**get_UponDisconnect**([out, retval] short * *pVal*);
HRESULT IHESession::**put_UponDisconnect**([in] short *newVal*);

Parameters

pVal—The following returned values specify the action that HostExplorer performs in the event that the host disconnects from the current session:

· UPONDISC_CLOSEWINDOW

· UPONDISC_KEEPWINDOWOPEN

· UPONDISC_RESTARTSESSION

· UPONDISC_SHOWOPENNEWSESSION

newVal—The set value, specifying the action that HostExplorer performs in the event that the host disconnects from the current session.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim iVal As Integer

iVal = Session.UponDisconnect

If (iVal <> UPONDISC_KEEPWINDOWOPEN) Then

Session.UponDisconnect = UPONDISC_KEEPWINDOWOPEN

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

short sVal;

pSess->get_UponDisconnect (&sVal);

if (sVal!= UPONDISC_KEEPWINDOWOPEN )

{

sVal= UPONDISC_KEEPWINDOWOPEN ;

pSess->put_UponDisconnect(sVal);

}
```

Property: IHESession::UseDialProp VT

This property returns or sets a value indicating whether HostExplorer uses the settings provided in the Windows Dialing Properties dialog box while you are installing a modem. When this property is set to VARIANT_TRUE, HostExplorer automatically adds the area code, country code, 8, 9, or calling-card numbers, if applicable. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = Session.**UseDialProp**

Session.**UseDialProp** = Boolean

C++ Syntax

HRESULT IHESession::**get_UseDialProp**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_UseDialProp**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer uses the settings provided in the Windows Dialing Properties dialog box. A returned value of VARIANT_FALSE indicates that HostExplorer dials using the “raw” phone number.

newVal—A value of VARIANT_TRUE indicates that HostExplorer uses the settings provided in the Windows Dialing Properties dialog box. A value of VARIANT_FALSE indicates that HostExplorer dials using the “raw” phone number.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.UseDialProp

If (bVal = False) Then

'Set value

Session.UseDialProp = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_UseDialProp (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_UseDialProp (bVal);

}
```

Property: IHESession::UserDirectory 3270 5250 VT

This property returns or sets a string specifying a user directory where objects such as Quick-Keys, key-map file, schemes, macros, and profiles are stored.

Basic Syntax

String = Session.**UserDirectory**
Session.**UserDirectory** = String

C++ Syntax

HRESULT IHESession::**get_UserDirectory**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_UserDirectory**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the user directory.
newVal—The set string, specifying the user directory.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

'Get string

strVal = Session.UserDirectory

If (Len(strVal) = 0) Then

'Set string

Session.UserDirectory = "C:\\Winnt\Hummingbird\7.00"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_UserDirectory (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\\Winnt\

Hummingbird\7.00"));

pSess->put_UserDirectory (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::UserName 5250

This property returns or sets a string that specifies the user name that you use to connect to the host.

Basic Syntax

String = Session.**UserName**
Session.**UserName** = String

C++ Syntax

HRESULT IHESession::**get_UserName**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_UserName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying your username.
newVal—The set string, specifying your username.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

'Get string

strVal = Session.UserName

If (Len(strVal) = 0) Then

'Set string

Session.UserName = "Jack17"

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

BSTR bstr ;

pSess->get_UserName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("Jack17"));

pSess->put_UserName (bstr);

SysFreeString(bstr);

}
```

Property: IHESession::VariableWidthFont 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer uses proportional fonts.

Basic Syntax

Boolean = Session.**VariableWidthFont**

Session.**VariableWidthFont** = Boolean

C++ Syntax

HRESULT IHESession::**get_VariableWidthFont**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_VariableWidthFont**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer uses proportional fonts. A returned value of VARIANT_FALSE indicates that HostExplorer does not use proportional fonts.

newVal—A value of VARIANT_TRUE indicates that HostExplorer uses proportional fonts. A value of VARIANT_FALSE indicates that HostExplorer does not use proportional fonts.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.VariableWidthFont

If (bVal = False) Then

'Set value

Session.VariableWidthFont = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_VariableWidthFont (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_VariableWidthFont (bVal);

}
```

Property: IHESession::VTCharset VT

This property returns a pointer to the SessionVTCharset interface to retrieve general options related to the character set.

Basic Syntax HESessionVTCharset = Session.**VTCharset**

C++ Syntax HRESULT IHESession::**get_VTCharset**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionVTCharset interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessVTChSet As HESessionVTCharset

Set SessVTChSet = Session.VTCharset

SessVTChSet.VTNRCMode = True
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionVTCharset * pVTCharset;

pSess->

get_VTCharset(&pVTCharset);

VARIANT_BOOL bVal = VARIANT_TRUE;

pVTCharset->

put_VTCharset (bVal);
```

Related Topics

TNVT UPSS Language-Conversion Table

Property: IHESession::VTDoHostWindowSize VT

This property returns or sets a value indicating whether HostExplorer sends or negotiates a change in window size (that is, the number of rows and columns to the Telnet host). By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = Session.**VTDoHostWindowSize**

Session.**VTDoHostWindowSize** = Boolean

C++ Syntax

HRESULT IHESession::**get_VTDoHostWindowSize**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESession::**put_VTDoHostWindowSize**([in] VARIANT_BOOL *newVal*);

Parameters

pVal— A returned value of VARIANT_TRUE indicates that HostExplorer sends or negotiates a change in window size. A returned value of VARIANT_FALSE indicates that HostExplorer does not send or negotiate a change in window size.

newVal—A value of VARIANT_TRUE indicates that HostExplorer sends or negotiates a change in window size. A value of VARIANT_FALSE indicates that HostExplorer does not send or negotiate a change in window size.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.VTDoHostWindowSize

If (bVal = False) Then

'Set value

Session.VTDoHostWindowSize = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_VTDoHostWindowSize (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_VTDoHostWindowSize (bVal);

}
```

Property: IHESession::VTInitiateTelnetNegotiation VT

This property returns or sets a value indicating whether HostExplorer (VT emulator) negotiates connection options when it establishes a Telnet connection. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = Session.**VTInitiateTelnetNegotiation**
Session.**VTInitiateTelnetNegotiation** = Boolean

C++ Syntax

HRESULT IHESession::**get_VTInitiateTelnetNegotiation**([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHESession::**put_VTInitiateTelnetNegotiation**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer negotiates connection options when it establishes a Telnet connection. A returned value of VARIANT_FALSE indicates that HostExplorer does not negotiate connection options when it establishes a Telnet connection.

newVal—A value of VARIANT_TRUE indicates that HostExplorer negotiates connection options when it establishes a Telnet connection. A value of VARIANT_FALSE indicates that HostExplorer does not negotiate connection options when it establishes a Telnet connection.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.VTInitiateTelnetNegotiation

If (bVal = False) Then

'Set value

Session.VTInitiateTelnetNegotiation=

True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_VTInitiateTelnetNegotiation (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->

put_VTInitiateTelnetNegotiation (bVal);

}
```

Property: IHESession::VTLineMode VT

This property returns or sets a value specifying how HostExplorer stores characters in a buffer until you send a carriage return to the host.

Basic Syntax

HOSTEX_LINEMODE = Session.**VTLineMode**

Session.**VTLineMode** = HOSTEX_LINEMODE

C++ Syntax

HRESULT IHESession::**get_VTLineMode**([out, retval] HOSTEX_LINEMODE * *pVal*);

HRESULT IHESession::**put_VTLineMode**([in] HOSTEX_LINEMODE *newVal*);

Parameters

pVal—The returned value specifying how HostExplorer stores characters in a buffer until you send a carriage return to the host.

newVal—The set value specifying how HostExplorer stores characters in a buffer until you send a carriage return to the host.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim LMode As HOSTEX_LINEMODE

LMode = Session.VTLineMode

If (LMode <> HOSTEX_LINEMODE_ALWAYS) Then

Session.VTLineMode = HOSTEX_LINEMODE_ALWAYS

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

HOSTEX_LINEMODE LMode;

pSess->get_VTLineMode (&TEcho);

if (LMode!= HOSTEX_LINEMODE_ALWAYS)

{

LMode = HOSTEX_LINEMODE_ALWAYS;

pSess->put_VTLineMode (LMode);

}
```

Related Topics

HOSTEX_LINEMODE Data Type

Property: IHESession::VTPrint VT

This property returns a pointer to the SessionVTPrint interface to retrieve the VT Printing options from the Output Group.

Basic Syntax HESessionVTPrint = Session.**VTPrint**

C++ Syntax HRESULT IHESession::**get_VTPrint**([out, retval] IUnknown * * *pVal*);

Parameters *pVal*—The returned pointer to the IHESessionVTPrint interface.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim SessVTPrint As HESessionVTPrint

Set SessVTPrint = Session.VTPrint

SessVTPrint.VTPrinterTimeout = 100
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

IHESessionVTPrint * pVTPrint;

pSess->get_VTPrint (&pVTPrint);

short sVal = 100;

pVTPrint->put_VTPrinterTimeout(&sVal);
```

Property: IHESession::WinDDEEnabled 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables or disables DDE (Dynamic Data Exchange). DDE allows programs to communicate with the emulator. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = Session.**WinDDEEnabled**

Session.**WinDDEEnabled** = Boolean

C++ Syntax

HRESULT IHESession::**get_WinDDEEnabled**([out, retval] VARIANT_BOOL *
pVal);

HRESULT IHESession::**put_WinDDEEnabled**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables DDE. A returned value of VARIANT_FALSE indicates that HostExplorer disables DDE.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables DDE. A value of VARIANT_FALSE indicates that HostExplorer disables DDE.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim bVal As Boolean

'Get value

bVal = Session.WinDDEEnabled

If (bVal = False) Then

'Set value

Session.WinDDEEnabled = True

End If
```

C++ Example

```
IDispatch *pIDispatch;

pITerminal->get_Session(&pIDispatch);

IHESession *pSess;

pIDispatch->QueryInterface(

IID_IHESession,

(void**) &pSess);

VARIANT_BOOL bVal;

bVal =

pSess->get_WinDDEEnabled &bVal);

if (bVal = VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSess->put_WinDDEEnabled (bVal);

}
```

Property: IHESession::WindowTitle 3270 5250 VT

This property returns or sets a string indicating the name and/or description displayed in the top right-hand corner of the session window.

Basic Syntax

String = Session.**WindowTitle**
Session.**WindowTitle** = String

C++ Syntax

HRESULT IHESession::**get_WindowTitle**([out, retval] BSTR * *pVal*);
HRESULT IHESession::**put_WindowTitle**([in] BSTR *newVal*);

Parameters

pVal—The returned string, which indicates the name and/or description.
newVal—The set string, which indicates the name and/or description.

Basic Example

```
Dim Session As HESession

Set Session = Terminal.Session

Dim strVal As String

'Get string

strVal = Session.WindowTitle

If (Len(strVal) = 0) Then

'Set string

Session.WindowTitle = "HostExplorer"

End If
```

C++ Example

```
IDispatch *pIDispatch;
pITerminal->get_Session(&pIDispatch);
IHESession *pSess;
pIDispatch->QueryInterface(
IID_IHESession,
(void**) &pSess);
BSTR bstr ;
pSess->get_WindowTitle (&bstr);
if (strlen(OLE2A(bstr))==0)
{
if (bstr!=NULL)
SysFreeString(bstr);
bstr =
SysAllocString(OLESTR("Host Explorer"));
pSess->put_WindowTitle (bstr);
SysFreeString(bstr);
}
```

SessionTerminal Interface

The SessionTerminal interface lets you set configuration settings related to the terminal.

Properties

The SessionTerminal interface consists of the following properties:

Cecp0	SpecialModelCols
Cecp1	SpecialModelRows
Cecp2	VT8BitMode
Cecp3	VTAnswerback
CharacterSet	VTAUPSS
ColumnSeparators	VTBSIsDel
ForceAltSize	VTConcealAnswerback
HostCharacterSet	VTDisplayMode
HostCodePage	VTForce8Bit
HostKeyboard	VTLocalEcho
Language	VTNewTerminalType
New3270EAB	VTOnLine
NewModel3279	VTScrollSpeed
NewModelType	VTSmoothScroll
ReplyOEM	VTTerminalID
SpecialModel	VTWrapLine

Property: IHESessionTerminal::Cecp0 3270

This property returns or sets a value that identifies the first byte of the CECP (Country Extended Code Page) to the host.

Basic Syntax

BYTE = SessionTerminal.**Cecp0**
SessionTerminal.**Cecp0** = BYTE

C++ Syntax

HRESULT IHESessionTerminal::**get_Cecp0**([out, retval] BYTE * *pVal*);
HRESULT IHESessionTerminal::**put_Cecp0**([in] BYTE *newVal*);

Parameters

pVal—The returned byte value (between 0 and 255).
newVal—The set byte value (between 0 and 255).

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim aByte As Byte

'Get value

aByte = SessTerm.Cecp0

If (aByte = 0) Then

'Set value

SessTerm.Cecp0 = 128

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BYTE aByte;

pTerm->get_Cecp0(&aByte);

if (aByte=0)

{

aByte=128;

pTerm->put_Cecp0(aByte);

}
```

Property: IHESessionTerminal::Cecp1 3270

This property returns or sets a value that identifies the second byte of the CECP (Country Extended Code Page) to the host.

Basic Syntax

BYTE = SessionTerminal.**Cecp1**
SessionTerminal.**Cecp1** = BYTE

C++ Syntax

HRESULT IHESessionTerminal::**get_Cecp1**([out, retval] BYTE * *pVal*);
HRESULT IHESessionTerminal::**put_Cecp1**([in] BYTE *newVal*);

Parameters

pVal—The returned byte value (between 0 and 255).
newVal—The set byte value (between 0 and 255).

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim aByte As Byte

'Get value

aByte = SessTerm.Cecp1

If (aByte = 0) Then

'Set value

SessTerm.Cecp1 = 128

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BYTE aByte;

pTerm->get_Cecp1(&aByte);

if (aByte=0)

{

aByte=128;

pTerm->put_Cecp1(aByte);

}
```

Property: IHESessionTerminal::Cecp2 3270

This property returns or sets a value that identifies the third byte of the CECP (Country Extended Code Page) to the host.

Basic Syntax

BYTE = SessionTerminal.**Cecp2**
SessionTerminal.**Cecp2** = BYTE

C++ Syntax

HRESULT IHESessionTerminal::**get_Cecp2**([out, retval] BYTE * *pVal*);
HRESULT IHESessionTerminal::**put_Cecp2**([in] BYTE *newVal*);

Parameters

pVal—The returned byte value (between 0 and 255).
newVal—The set byte value (between 0 and 255).

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim aByte As Byte

'Get value

aByte = SessTerm.Cecp2

If (aByte = 0) Then

'Set value

SessTerm.Cecp2 = 128

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BYTE aByte;

pTerm->get_Cecp2(&aByte);

if (aByte=0)

{

aByte=128;

pTerm->put_Cecp2(aByte);

}
```

Property: IHESessionTerminal::Cecp3 3270

This property returns or sets a value that identifies the fourth byte of the CECP (Country Extended Code Page) to the host.

Basic Syntax

```
BYTE = SessionTerminal.Cecp3
SessionTerminal.Cecp3 = BYTE
```

C++ Syntax

```
HRESULT IHESessionTerminal::get_Cecp3([out, retval] BYTE * pVal);
HRESULT IHESessionTerminal::put_Cecp3([in] BYTE newVal);
```

Parameters

pVal—The returned byte value (between 0 and 255).
newVal—The set byte value (between 0 and 255).

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim aByte As Byte

'Get value

aByte = SessTerm.Cecp3

If (aByte = 0) Then

'Set value

SessTerm.Cecp3 = 128

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BYTE aByte;

pTerm->get_Cecp3(&aByte);

if (aByte=0)

{

aByte=128;

pTerm->put_Cecp3(aByte);

}
```

Property: IHESessionTerminal::CharacterSet 5250

This property returns or sets a value that specifies character-set information for the host.

Basic Syntax

Integer = SessionTerminal.**CharacterSet**
SessionTerminal.**CharacterSet** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_CharacterSet**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_CharacterSet**([in] short *newVal*);

Parameters

pVal—The returned string, which indicates the character-set information.
newVal—The set string, which indicates the character-set information.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.CharacterSet

If (iVal = 0) Then

'Set value

SessTerm.CharacterSet = 3

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_CharacterSet(&sVal);

if (sVal==0)

{

sVal = 3;

pTerm->put_CharacterSet(sVal);

}
```

Related Topics

TN5250 Language-Conversion Table

Property: IHESessionTerminal::ColumnSeparators 5250

This property returns or sets a value specifying the column-separator style for the session. By default, this property is set to Dots.

Basic Syntax

Integer = SessionTerminal.**ColumnSeparators**
SessionTerminal.**ColumnSeparators** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_ColumnSeparators**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_ColumnSeparators**([in] short *newVal*);

Parameters

pVal—A returned value of:

· None—Indicates that HostExplorer does not display separators between columns.

· Dots—Indicates that HostExplorer displays column separators as dots on the terminal window.

· Lines—Indicates that HostExplorer displays column separators as lines on the terminal window.

newVal—A set value of:

· None—Indicates that HostExplorer does not display separators between columns.

· Dots—Indicates that HostExplorer displays column separators as dots on the terminal window.

· Lines—Indicates that HostExplorer displays column separators as lines on the terminal window.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.ColumnSeparators

If (iVal = 0) Then

'Set value

SessTerm.ColumnSeparators = 3

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_ColumnSeparators (&sVal);

if (sVal==0)

{

sVal = 3;

pTerm->put_ColumnSeparators (sVal);

}
```

Property: IHESessionTerminal::ForceAltSize 3270

This property returns or sets a value indicating whether HostExplorer changes the window to the alternate size when the host receives an Erase Write command. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionTerminal.**ForceAltSize**

SessionTerminal.**ForceAltSize** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_ForceAltSize**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_ForceAltSize**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer changes the window to the alternate size. A returned value of VARIANT_FALSE indicates that HostExplorer does not change the window to the alternate size.

newVal—A value of VARIANT_TRUE indicates that HostExplorer changes the window to the alternate size. A value of VARIANT_FALSE indicates that HostExplorer does not change the window to the alternate size.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.ForceAltSize

If (bVal = False) Then

'Set value

SessTerm.ForceAltSize = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_ForceAltSize(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_ForceAltSize(bVal);

}
```

Property: IHESessionTerminal::HostCharacterSet 5250

This property indicates the CSID5250 (Character Set ID for the 5250 terminal) parameter necessary for negotiating with the host.

Basic Syntax

String = SessionTerminal.**HostCharacterSet**
SessionTerminal.**HostCharacterSet** = String

C++ Syntax

HRESULT IHESessionTerminal::**get_HostCharacterSet**([out, retval] BSTR * *pVal*);
HRESULT IHESessionTerminal::**put_HostCharacterSet**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the host character set.
newVal—The set string, specifying the host character set.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim strVal As String

'Get string

strVal = SessTerm.HostCharacterSet

If (Len(strVal) = 0) Then

'Set string

SessTerm.HostCharacterSet = "00697"

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BSTR bstr;

pTerm->get_HostCharacterSet(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("00697"));

pTerm->put_HostCharacterSet(bstr);

SysFreeString(bstr);

}
```

Related Topics

TN5250 Language-Conversion Table

Property: IHESessionTerminal::HostCodePage 5250

This property returns or sets a value indicating the HostCodePage parameter necessary for negotiating with the host.

Basic Syntax

String = SessionTerminal.**HostCodePage**
SessionTerminal.**HostCodePage** = String

C++ Syntax

HRESULT IHESessionTerminal::**get_HostCodePage**([out, retval] BSTR * *pVal*);
HRESULT IHESessionTerminal::**put_HostCodePage**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the host code page.
newVal—The set string, specifying the host code page.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim strVal As String

'Get string

strVal = SessTerm.HostCodePage

If (Len(strVal) = 0) Then

'Set string

SessTerm.HostCodePage = "00037"

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BSTR bstr;

pTerm->get_HostCodePage (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("00037"));

pTerm->put_HostCodePage (bstr);

SysFreeString(bstr);

}
```

Related Topics

TN5250 Language-Conversion Table

Property: IHESessionTerminal::HostKeyboard 5250

This property indicates the KBDID5250 (Keyboard ID for the 5250 terminal) parameter necessary for negotiating with the host.

Basic Syntax

String = SessionTerminal.**HostKeyboard**
SessionTerminal.**HostKeyboard** = String

C++ Syntax

HRESULT IHESessionTerminal::**get_HostKeyboard**([out, retval] BSTR * *pVal*);
HRESULT IHESessionTerminal::**put_HostKeyboard**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the host keyboard.
newVal—The set string, specifying the host keyboard.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim strVal As String

'Get string

strVal = SessTerm.HostKeyboard

If (Len(strVal) = 0) Then

'Set string

SessTerm.HostKeyboard = "USB"

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BSTR bstr;

pTerm->get_HostKeyboard (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("USB"));

pTerm->put_HostKeyboard (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionTerminal::Language VT

This property returns or sets the host code page that you want to use. It influences the EBCDIC-to-ASCII or EBCDIC-to-Unicode conversion table.

Basic Syntax

String = SessionTerminal.**Language**
SessionTerminal.**Language** = String

C++ Syntax

HRESULT IHESessionTerminal::**get_Language**([out, retval] BSTR * *pVal*);
HRESULT IHESessionTerminal::**put_Language**([in] BSTR *newVal*);

Parameters

pVal—The returned value of the host code page.
newVal—The set value of the host code page.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim strVal As String

'Get string

strVal = SessTerm.Language

If (Len(strVal) = 0) Then

'Set string

SessTerm.Language = "ISO Latin-9"

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BSTR bstr;

pTerm->get_Language (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("USB"));

pTerm->put_Language(bstr);

SysFreeString(bstr);

}
```

Related Topics

TNVT UPSS Language-Conversion Table

Property: IHESessionTerminal::New3270EAB 3270

This property returns or sets a value indicating whether HostExplorer enables Extended Attributes when you select 3279 as the 3270 type. Extended Attributes are the mainframe application codes used to display various colors, highlighting, reverse images, and blinking; they let your computer fully emulate the mainframe screen. The change in this value takes effect only after you disconnect and reconnect the session. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionTerminal.**New3270EAB**

SessionTerminal.**New3270EAB** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_New3270EAB**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_New3270EAB**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables Extended Attributes. A returned value of VARIANT_FALSE indicates that HostExplorer disables Extended Attributes.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables Extended Attributes. A value of VARIANT_FALSE indicates that HostExplorer disables Extended Attributes.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.New3270EAB

If (bVal = False) Then

'Set value

SessTerm. New3270EAB = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_New3270EAB (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_New3270EAB (bVal);

}
```

Property: IHESessionTerminal::NewModel3279 3270

This property returns or sets a value indicating whether HostExplorer supports Extended Attributes mode. By default, this property is set to TRUE.

Basic Syntax

SessionTerminal.**NewModel3279** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_NewModel3279**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_NewModel3279**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE (3279) indicates that HostExplorer supports Extended Attributes mode. A returned value of VARIANT_FALSE (3278) indicates that HostExplorer does not support Extended Attributes mode.

newVal—A value of VARIANT_TRUE (3279) indicates that HostExplorer supports Extended Attributes mode. A value of VARIANT_FALSE (3278) indicates that HostExplorer does not support Extended Attributes mode.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.NewModel3279

If (bVal = False) Then

'Set value

SessTerm.NewModel3279 = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_NewModel3279 (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_NewModel3279 (bVal);

}
```

Property: IHESessionTerminal::NewModelType 3270 5250 VT

This property returns or sets a value (Short integer) specifying the model type for the session. After the terminal model is changed, you must disconnect from and reconnect to the host. Otherwise, HostExplorer does not update the change to the current session. Changing this value takes effect only after you disconnect and reconnect the session. VT220 is the most commonly supported terminal on most UNIX systems. IBM 3151 is used to connect to AIX systems.

Note: Use the VT320 or VT420 terminal type only if you have proper termcap entries on the host. Use SCO ANSI when connecting to SCO UNIX systems.

Basic Syntax

Integer = SessionTerminal.**NewModelType**
SessionTerminal.**NewModelType** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_NewModelType**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_NewModelType**([in] short *newVal*);

Parameters

pVal—For TN3270 terminals, the following returned values specify the model type for the session.

· 2—Model 2 (24x80)

· 3—Model 3 (32x80)

· 4—Model 4 (43x80)

· 5—Model 5 (27x132)

For TN5250 terminals, the following returned values specify the model type for the session:

· 2—Model 2 (24x80)

· 5—Model 5 (27x132)

For TNVT terminals, the following returned values specify the model type for the session:

· HOSTEX_TERM_MODEL_VT52

· HOSTEX_TERM_MODEL_VT100

· HOSTEX_TERM_MODEL_VT101

· HOSTEX_TERM_MODEL_VT102

· HOSTEX_TERM_MODEL_VT220

· HOSTEX_TERM_MODEL_VT320

· HOSTEX_TERM_MODEL_VT420

· HOSTEX_TERM_MODEL_ANSI

· HOSTEX_TERM_MODEL_SCOANSI

· HOSTEX_TERM_MODEL_TERMIBM3151

newVal—The set value, specifying the model type for the session.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.NewModelType

If (iVal = 0) Then

'Set value

SessTerm.NewModelType = 6

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_NewModelType(&sVal);

if (sVal==0)

{

sVal = 6;

pTerm->put_NewModelType(sVal);

}
```

Property: IHESessionTerminal::ReplyOEM 3270

This property returns or sets a value indicating whether HostExplorer sends an OEM reply field back to the host in response to receiving a Read Partition Query. If sent, the OEM reply contains information about the terminal session and features available for the host to use. Clear this option if you experience difficulty starting GDDM, SAS, or other mainframe applications. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionTerminal.**ReplyOEM**

SessionTerminal.**ReplyOEM** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_ReplyOEM**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_ReplyOEM**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer sends an OEM reply field back to the host. A returned value of VARIANT_FALSE indicates that HostExplorer sends an OEM reply field back to the host.

newVal—A value of VARIANT_TRUE indicates that HostExplorer sends an OEM reply field back to the host. A value of VARIANT_FALSE indicates that HostExplorer does not send an OEM reply field back to the host.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.ReplyOEM

If (bVal = False) Then

'Set value

SessTerm.ReplyOEM = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_ReplyOEM(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_ReplyOEM(bVal);

}
```

Property: IHESessionTerminal::SpecialModel 3270

This property returns or sets a value indicating whether HostExplorer is running a special model of the mainframe terminal to emulate the session.

Basic Syntax

Boolean = SessionTerminal.**SpecialModel**

SessionTerminal.**SpecialModel** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_SpecialModel**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_SpecialModel**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer is running a special model of the mainframe. A returned value of VARIANT_FALSE indicates that HostExplorer is not running a special model of the mainframe.

newVal—A value of VARIANT_TRUE indicates that HostExplorer is running a special model of the mainframe. A value of VARIANT_FALSE indicates that HostExplorer is not running a special model of the mainframe.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.SpecialModel

If (bVal = False) Then

'Set value

SessTerm.SpecialModel = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_SpecialModel (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_SpecialModel (bVal);

}
```

Property: IHESessionTerminal::SpecialModelCols 3270 5250 VT

This property returns or sets a value (Short integer) specifying the special model number of columns for the session.

Basic Syntax

Integer = SessionTerminal.**SpecialModelCols**
SessionTerminal.**SpecialModelCols** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_SpecialModelCols**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_SpecialModelCols**([in] short *newVal*);

Parameters

pVal—The returned value, specifying the special model number of columns for the session.
newVal—The set value, specifying the special model number of columns for the session.

Basic Example

```
Dim iVal As Integer

'Get value

iVal = SessTerm.SpecialModelCols

If (iVal = 0) Then

'Set value

SessTerm.SpecialModelCols = 5

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_SpecialModelCols(&sVal);

if (sVal==0)

{

sVal = 5;

pTerm->put_SpecialModelCols(sVal);

}
```

Property: IHESessionTerminal::SpecialModelRows 3270 5250 VT

This property returns or sets a value specifying the special model number of rows for the session.

Basic Syntax

Integer = SessionTerminal.**SpecialModelRows**
SessionTerminal.**SpecialModelRows** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_SpecialModelRows**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_SpecialModelRows**([in] short *newVal*);

Parameters

pVal—The returned value, specifying the special model number of rows for the session.
newVal—The set value, specifying the special model number of rows for the session.

Basic Example

```
Dim iVal As Integer

'Get value

iVal = SessTerm.SpecialModelRows

If (iVal = 0) Then

'Set value

SessTerm.SpecialModelRows = 5

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_SpecialModelRows (&sVal);

if (sVal==0)

{

sVal = 5;

pTerm->put_SpecialModelRows (sVal);

}
```

Property: IHESessionTerminal::VT8BitMode VT

This property returns or sets a value indicating whether HostExplorer uses the 8-bit transmission mode to connect to the host. This 8-bit mode supports 7-bit and 8-bit data formats.

Basic Syntax

Boolean = SessionTerminal.**VT8BitMode**
SessionTerminal.**VT8BitMode** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VT8BitMode**([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHESessionTerminal::**put_VT8BitMode**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer uses the 8-bit transmission mode. A returned value of VARIANT_FALSE indicates that HostExplorer does not use the 8-bit transmission mode.
newVal—A value of VARIANT_TRUE indicates that HostExplorer uses the 8-bit transmission mode. A value of VARIANT_FALSE indicates that HostExplorer does not use the 8-bit transmission mode.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VT8BitMode

If (bVal = False) Then

'Set value

SessTerm.VT8BitMode = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VT8BitMode (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VT8BitMode (bVal);

}
```

Property: IHESessionTerminal::VTAnswerback VT

This property returns or sets a string specifying an Answerback message. You can enter special-character sequences in this field.

Basic Syntax

String = SessionTerminal.**VTAnswerback**
SessionTerminal.**VTAnswerback** = String

C++ Syntax

HRESULT IHESessionTerminal::**get_VTAnswerback**([out, retval] BSTR * *pVal*);
HRESULT IHESessionTerminal::**put_VTAnswerback**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the Answerback message.
newVal—The set string, specifying the Answerback message.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim strVal As String

'Get string

strVal = SessTerm.VTAnswerback

If (Len(strVal) = 0) Then

'Set string

SessTerm.VTAnswerback = "ls -l"

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

BSTR bstr;

pTerm->get_VTAnswerback (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("ls -l"));

pTerm->put_VTAnswerback (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionTerminal::VTAUPSS VT

This property returns or sets an integer, which corresponds to the language of the terminal.

Basic Syntax

```
Integer = SessionTerminal.VTAUPSS
SessionTerminal.VTAUPSS = Integer
```

C++ Syntax

```
HRESULT IHESessionTerminal::get_VTAUPSS([out, retval] short * pVal);
HRESULT IHESessionTerminal::put_VTAUPSS([in] short newVal);
```

Parameters

pVal—The returned integer, which indicates the language of the terminal.
newVal—The set value, which indicates the language of the terminal.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.VTAUPSS

If (iVal = 0) Then

'Set value. The number 123 corresponds

‘to hex 0x007B, which corresponds to

‘ISO Latin-1

SessTerm.VTAUPSS = 123

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_VTAUPSS(&sVal);

if (sVal==0)

{

sVal = 123;

pTerm->put_VTAUPSS (sVal);

}
```

Related Topics

TNVT UPSS Language-Conversion Table

Property: IHESessionTerminal::VTBSIsDel VT

This property returns or sets a value indicating the type of code that the Backspace key sends.

Basic Syntax

Boolean = SessionTerminal.**VTBSIsDel**

SessionTerminal.**VTBSIsDel** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VTBSIsDel**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_VTBSIsDel**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Backspace key sends a Del (127) code. A returned value of VARIANT_FALSE indicates that the Backspace key sends a BS (8) code.

newVal—A value of VARIANT_TRUE indicates that the Backspace key sends a Del (127) code. A value of VARIANT_FALSE indicates that the Backspace key sends a BS (8) code.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VTBSIsDel

If (bVal = False) Then

'Set value

SessTerm.VTBSIsDel = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VTBSIsDel (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VTBSIsDel (bVal);

}
```

Property: IHESessionTerminal::VTConcealAnswerback VT

This property returns or sets a value indicating whether HostExplorer clears the Answerback message field. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionTerminal.**VTConcealAnswerback**

SessionTerminal.**VTConcealAnswerback** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VTConcealAnswerback**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_VTConcealAnswerback**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer clears the Answerback message field. A returned value of VARIANT_FALSE indicates that HostExplorer does not clear the Answerback message field.

newVal—A value of VARIANT_TRUE indicates that HostExplorer clears the Answerback message field. A value of VARIANT_FALSE indicates that HostExplorer does not clear the Answerback message field.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VTConcealAnswerback

If (bVal = False) Then

'Set value

SessTerm.VTConcealAnswerback = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VTConcealAnswerback (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VTConcealAnswerback (bVal);

}
```

Property: IHESessionTerminal::VTDisplayMode VT

This property returns or sets a value specifying the display mode. By default, this property is set to Optimized mode.

Basic Syntax

Integer = SessionTerminal.**VTDisplayMode**

SessionTerminal.**VTDisplayMode** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_VTDisplayMode**([out, retval] short * *pVal*);

HRESULT IHESessionTerminal::**put_VTDisplayMode**([in] short *newVal*);

Parameters

pVal—A returned value of Optimized mode indicates that HostExplorer performs “bulk” updates to the screen. Typically, the emulator performs bulk updates at the end of a data stream. A returned value of Realistic mode indicates that HostExplorer updates the screen as it receives new characters. Although this is a much slower option, it allows for smoother scrolling.

newVal—A value of Optimized mode indicates that HostExplorer performs “bulk” updates to the screen. Typically, the emulator performs bulk updates at the end of a data stream. A value of Realistic mode indicates that HostExplorer updates the screen as it receives new characters.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.VTDisplayMode

If (iVal = 0) Then

'Set value

SessTerm.VTDisplayMode = 1

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_VTDisplayMode(&sVal);

if (sVal==0)

{

sVal = 1;

pTerm->put_VTDisplayMode(sVal);

}
```

Property: IHESessionTerminal::VTForce8Bit VT

This property returns or sets a value indicating whether HostExplorer supports 8-bit data transfers even when NRC support is enabled. Usually the high-order bit of incoming data is stripped when NRC is enabled. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionTerminal.**VTForce8Bit**

SessionTerminal.**VTForce8Bit** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VTForce8Bit**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_VTForce8Bit**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer supports 8-bit data transfers. A returned value of VARIANT_FALSE indicates that HostExplorer does not support 8-bit data transfers.

newVal—A value of VARIANT_TRUE indicates that HostExplorer supports 8-bit data transfers. A value of VARIANT_FALSE indicates that HostExplorer does not support 8-bit data transfers.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VTForce8Bit

If (bVal = False) Then

'Set value

SessTerm.VTForce8Bit = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VTForce8Bit (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VTForce8Bit (bVal);

}
```

Property: IHESessionTerminal::VTLocalEcho VT

This property returns or sets a value indicating whether HostExplorer enables local echo of characters that you type in the emulator. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionTerminal.**VTLocalEcho**

SessionTerminal.**VTLocalEcho** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VTLocalEcho**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_VTLocalEcho**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer echoes all the data that you type. A returned value of VARIANT_FALSE indicates that HostExplorer does not echo any data that you type.

newVal—A value of VARIANT_TRUE indicates that HostExplorer echoes all the data that you type. A value of VARIANT_FALSE indicates that HostExplorer does not echo any data that you type.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VTLocalEcho

If (bVal = False) Then

'Set value

SessTerm.VTLocalEcho = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VTLocalEcho (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VTLocalEcho (bVal);

}
```

Property: IHESessionTerminal::VTNewTerminalType VT

This property returns or sets a value specifying the VT terminal type to use for the current session. After changing the terminal type, you must disconnect and reconnect to the host. Otherwise, HostExplorer does not update the the current session. VT220 is the most commonly supported terminal on most UNIX systems.

Note: Use VT320 or VT420 only if you have proper termcap entries on the host. Use SCO ANSI when connecting to SCO UNIX systems.

Basic Syntax

Integer = SessionTerminal.**VTNewTerminalType**
SessionTerminal.**VTNewTerminalType** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_VTNewTerminalType**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_VTNewTerminalType**([in] short *newVal*);

Parameters

pVal—The following returned values specify the VT terminal type to use for the current session:

· 2—MODEL 2

· 3—MODEL 3

· 4—MODEL 4

· 5—MODEL 5

· 6—VT52

· 7—VT100

· 8—VT101

· 9—VT102

· 10—VT220

· 11—VT320

· 12—VT420

· 13—ANSI

· 14—SCOANSI

· 15—TERM_IBM3151

· 16—WYSE50

· 17—WYSE60

newVal—The set value, specifying the VT terminal type to use for the current session.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.VTNewTerminalType

If (iVal = 5) Then

'Set value

SessTerm. VTNewTerminalType = 9

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_VTNewTerminalType (&sVal);

if (sVal==0)

{

sVal = 9;

pTerm->put_VTNewTerminalType (sVal);

}
```

Property: IHESessionTerminal::VTOnLine VT

This property returns or sets a value indicating whether HostExplorer sends data to the host. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionTerminal.**VTOnLine**

SessionTerminal.**VTOnLine** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VTOnLine**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_VTOnLine**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the terminal is online and sends data to the host. A returned value of VARIANT_FALSE indicates that the terminal is offline, so you can type and move the cursor around the screen without sending data to the host.

newVal—A value of VARIANT_TRUE indicates that the terminal is online and sends data to the host. A value of VARIANT_FALSE indicates that the terminal is offline, so you can type and move the cursor around the screen without sending data to the host.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VTOnLine

If (bVal = False) Then

'Set value

SessTerm.VTOnLine = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VTOnLine (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VTOnLine (bVal);

}
```

Property: IHESessionTerminal::VTScrollSpeed VT

This property returns or sets a value specifying the Smooth Scrolling speed. The value can range from 1 to 30. If the value is greater than the number of vertical pixels per cell, scrolling occurs in Line mode.

Basic Syntax

Integer = SessionTerminal.**VTScrollSpeed**
SessionTerminal.**VTScrollSpeed** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_VTScrollSpeed**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_VTScrollSpeed**([in] short *newVal*);

Parameters

pVal—The returned value, specifying the Smooth Scrolling speed. The value can range from 2 to the value of the font height.
newVal—The set value, specifying the Smooth Scrolling speed. The value can range from 2 to the value of the font height.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.VTScrollSpeed

If (iVal = 0) Then

'Set value

SessTerm.VTScrollSpeed = 4

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_VTScrollSpeed (&sVal);

if (sVal==0)

{

sVal = 4;

pTerm->put_VTScrollSpeed (sVal);

}
```

Property: IHESessionTerminal::VTSmoothScroll VT

This property returns or sets a value indicating whether HostExplorer uses the Smooth Scrolling feature. Before setting this property, you have to set the VT Display mode to Realistic.

Basic Syntax

Boolean = SessionTerminal.**VTSmoothScroll**

SessionTerminal.**VTSmoothScroll** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VTSmoothScroll**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_VTSmoothScroll**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer scrolls data using a smooth scroll method. A returned value of VARIANT_FALSE indicates that HostExplorer does not scroll data line by line.

newVal—A value of VARIANT_TRUE indicates that HostExplorer scrolls data using a smooth scroll method. A value of VARIANT_FALSE indicates that HostExplorer does not scroll data line by line.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VTSmoothScroll

If (bVal = False) Then

'Set value

SessTerm.VTSmoothScroll = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VTSmoothScroll (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VTSmoothScroll (bVal);

}
```

Property: IHESessionTerminal::VTTerminalID VT

This property returns or sets a value specifying the terminal ID or Device Attribute (DA) response that HostExplorer sends to the host. The DA contains the control sequences that define the terminal and its configuration and identifies the particular type of terminal to the host.

Basic Syntax

Integer = SessionTerminal.**VTTerminalID**
SessionTerminal.**VTTerminalID** = Integer

C++ Syntax

HRESULT IHESessionTerminal::**get_VTTerminalID**([out, retval] short * *pVal*);
HRESULT IHESessionTerminal::**put_VTTerminalID**([in] short *newVal*);

Parameters

pVal—The following returned values specify the terminal ID or DA response that HostExplorer sends to the host:

· 0—VT100

· 1—VT101

· 2—VT102

· 3—VT220

· 4—VT320

· 5—VT420

· 6—VT80

· 7—VT100J

· 8—VT102J

· 9—VT220J

· 10—VT282

· 11—VT382

newVal—The set value, specifying the terminal ID or DA response that HostExplorer sends to the host.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim iVal As Integer

'Get value

iVal = SessTerm.VTTerminalID

If (iVal = 0) Then

'Set value

SessTerm.VTTerminalID = 6

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

short sVal;

pTerm->get_VTTerminalID (&sVal);

if (sVal==0)

{

sVal = 6;

pTerm->put_VTTerminalID (sVal);

}
```

Property: IHESessionTerminal::VTWrapLine VT

This property returns or sets a value indicating whether HostExplorer automatically wraps lines that extend past the last column on the screen. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionTerminal.**VTWrapLine**

SessionTerminal.**VTWrapLine** = Boolean

C++ Syntax

HRESULT IHESessionTerminal::**get_VTWrapLine**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionTerminal::**put_VTWrapLine**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically wraps lines. A returned value of VARIANT_FALSE indicates that HostExplorer does not automatically wrap lines.

newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically wraps lines. A value of VARIANT_FALSE indicates that HostExplorer does not automatically wrap lines.

Basic Example

```
Dim SessTerm As HESessionTerminal

Set SessTerm = Session.Terminal

Dim bVal As Boolean

'Get value

bVal = SessTerm.VTWrapLine

If (bVal = False) Then

'Set value

SessTerm.VTWrapLine = True

End If
```

C++ Example

```
IHESessionTerminal *pTerm;

pSess->get_Terminal(&pTerm);

VARIANT_BOOL bVal;

pTerm->get_VTWrapLine (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pTerm->put_VTWrapLine (bVal);

}
```

SessionGraphics Interface

The SessionGraphics interface lets you set configuration settings related to graphics.

Properties

The SessionGraphics interface consists of the following properties:

APL

GraphicsCursorType

GraphicsModel

LightPen

ProgramSymbols

PSCellSize

Property: IHESessionGraphics::APL 3270

This property returns or sets a value that indicates whether HostExplorer supports APL (A Program Language). By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionGraphics.**APL**

SessionGraphics.**APL** = Boolean

C++ Syntax

HRESULT IHESessionGraphics::**get_APL**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionGraphics::**put_APL**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer supports APL. A returned value of VARIANT_FALSE indicates that HostExplorer does not support APL.

newVal—A value of VARIANT_TRUE indicates that HostExplorer supports APL. A value of VARIANT_FALSE indicates that HostExplorer does not support APL.

Basic Example

```
Dim SessGraph As HESessionGraphics

Set SessGraph = Session.Graphics

Dim bVal As Boolean

'Get value

bVal = SessGraph.APL

If (bVal = False) Then

'Set value

SessGraph.APL = True

End If
```

C++ Example

```
IHESessionGraphics* pGraphics;

pSess->get_Graphics(&pGraphics);

VARIANT_BOOL bVal;

pGraphics->get_APL(&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pGraphics->put_APL(bVal);

}
```

Property: IHESessionGraphics::GraphicsCursorType 3270

This property lets you determine how the cursor appears in the terminal window.

Basic Syntax

HOSTEX_GRAPHICS_CURSOR_TYPE = SessionGraphics.**GraphicsCursorType**

SessionGraphics.**GraphicsCursorType** = HOSTEX_GRAPHICS_CURSOR_TYPE

C++ Syntax

HRESULT IHESessionGraphics::**get_GraphicsCursorType**([out, retval] HOSTEX_GRAPHICS_CURSOR_TYPE * *pVal*);

HRESULT IHESessionGraphics::**put_GraphicsCursorType**([in] HOSTEX_GRAPHICS_CURSOR_TYPE *newVal*);

Parameters

pVal—The returned value, which indicates how the cursor appears in the terminal window.

newVal—The set value, which indicates how the cursor appears in the terminal window.

Basic Example

```
Dim SessGraph As HESessionGraphics

Set SessGraph = Session.Graphics

Dim CurType As HOSTEX_GRAPHICS_CURSOR_TYPE

CurType = SessGraph.GraphicsCursorType

If (CurType <> HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE)

Then

SessGraph.GraphicsCursorType =

HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE

End If
```

C++ Example

```
IHESessionGraphics* pGraphics;

pSess->get_Graphics(&pGraphics);

HOSTEX_GRAPHICS_CURSOR_TYPE CurType;

pGraphics->

get_GraphicsCursorType(&CurType);

if (CurType!= HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE)

{

CurType =

HOSTEX_GRAPHICS_CURSOR_TYPE_LARGE_CROSS_WHITE;

pGraphics->

put_GraphicsCursorType (CurType);

}
```

Related Topics

HOSTEX_GRAPHICS_CURSOR_TYPE Data Type

Property: IHESessionGraphics::GraphicsModel 3270

This property lets you select the graphics terminal model to use during the next session.

Basic Syntax

HOSTEX_GRAPHICS_MODEL = SessionGraphics.**GraphicsModel**

SessionGraphics.**GraphicsModel** = HOSTEX_GRAPHICS_MODEL

C++ Syntax

HRESULT IHESessionGraphics::**get_GraphicsModel**([out, retval] HOSTEX_GRAPHICS_MODEL * *pVal*);

HRESULT IHESessionGraphics::**put_GraphicsModel**([in] HOSTEX_GRAPHICS_MODEL *newVal*);

Parameters

pVal—The returned value, which indicates the graphics terminal model to use during the next session.

newVal—The set value, which indicates the graphics terminal model to use during the next session.

Basic Example

```
Dim SessGraph As HESessionGraphics

Set SessGraph = Session.Graphics

Dim GraphMod As HOSTEX_GRAPHICS_MODEL

GraphMod = SessGraph.GraphicsModel

If (GraphMod <>

HOSTEX_GRAPHICS_MODEL_3179G)

Then

SessGraph.GraphicsModel =

HOSTEX_GRAPHICS_MODEL_3179G

End If
```

C++ Example

```
IHESessionGraphics* pGraphics;

pSess->get_Graphics(&pGraphics);

HOSTEX_GRAPHICS_MODEL GraphMod;

pGraphics->get_GraphicsModel(&GraphMod);

if (GraphMod!=HOSTEX_GRAPHICS_MODEL_3179G)

{

GraphMod = HOSTEX_GRAPHICS_MODEL_3179G;

pGraphics->

put_GraphicsModel (GraphMod);

}
```

Related Topics

HOSTEX_GRAPHICS_MODEL Data Type

Property: IHESessionGraphics::LightPen 3270

This property returns or sets a value indicating whether HostExplorer allows you to use the keyboard and/or mouse to emulate light-pen functionality. This functionality allows you to use a light-sensitive device to select screen fields. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionGraphics.**LightPen**

SessionGraphics.**LightPen** = Boolean

C++ Syntax

HRESULT IHESessionGraphics::**get_LightPen**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionGraphics::**put_LightPen**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer allows you to use the keyboard and/or mouse to emulate light-pen functionality. A returned value of VARIANT_FALSE indicates that HostExplorer does not allow you to use the keyboard and/or mouse to emulate light-pen functionality.

newVal—A value of VARIANT_TRUE indicates that HostExplorer allows you to use the keyboard and/or mouse to emulate light-pen functionality. A value of VARIANT_FALSE indicates that HostExplorer does not allow you to use the keyboard and/or mouse to emulate light-pen functionality.

Basic Example

```
Dim SessGraph As HESessionGraphics

Set SessGraph = Session.Graphics

Dim bVal As Boolean

'Get value

bVal = SessGraph.LightPen

If (bVal = False) Then

'Set value

SessGraph.LightPen = True

End If
```

C++ Example

```
IHESessionGraphics* pGraphics;

pSess->get_Graphics(&pGraphics);

VARIANT_BOOL bVal;

pGraphics->get_LightPen(&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pGraphics->put_LightPen(bVal);

}
```

Property: IHESessionGraphics::ProgramSymbols 3270

This property lets you determine whether HostExplorer supports program symbols. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionGraphics.**ProgramSymbols**

SessionGraphics.**ProgramSymbols** = Boolean

C++ Syntax

HRESULT IHESessionGraphics::**get_ProgramSymbols**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionGraphics::**put_ProgramSymbols**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer supports program symbols. A returned value of VARIANT_FALSE indicates that HostExplorer does not support program symbols.

newVal—A set value of VARIANT_TRUE indicates that HostExplorer supports program symbols. A set value of VARIANT_FALSE indicates that HostExplorer does not support program symbols.

Basic Example

```
Dim SessGraph As HESessionGraphics

Set SessGraph = Session.Graphics

Dim bVal As Boolean

'Get value

bVal = SessGraph.ProgramSymbols

If (bVal = False) Then

'Set value

SessGraph.ProgramSymbols = True

End If
```

C++ Example

```
IHESessionGraphics* pGraphics;

pSess->get_Graphics(&pGraphics);

VARIANT_BOOL bVal;

pGraphics->get_ProgramSymbols (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pGraphics->put_ProgramSymbols (bVal);

}
```

Property: IHESessionGraphics::PSCellSize 3270

This property returns or sets a value that determines the cell size when HostExplorer reports a fixed-coordinate space to the host. Because several host systems are sensitive to cell size, you need to select the appropriate size. By default, this property is set to HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC.

Basic Syntax

HOSTEX_GRAPHICS_CELL_SIZE = SessionGraphics.**PSCellSize**

SessionGraphics.**PSCellSize** = HOSTEX_GRAPHICS_CELLSIZE

C++ Syntax

HRESULT IHESessionGraphics::**get_PSCellSize**([out, retval] HOSTEX_GRAPHICS_CELLSIZE * *pVal*);

HRESULT IHESessionGraphics::**put_PSCellSize**([in] HOSTEX_GRAPHICS_CELLSIZE *newVal*);

Parameters

pVal—The returned value, which determines the cell size when HostExplorer reports a fixed-coordinate space to the host.

newVal—The set value, which determines the cell size when HostExplorer reports a fixed-coordinate space to the host.

Basic Example

```
Dim SessGraph As HESessionGraphics
Set SessGraph = Session.Graphics
Dim CellSize As HOSTEX_GRAPHICS_CELL_SIZE
CellSize = SessGraph.PSCellSize
If (CellSize <>
HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC)
Then
SessGraph.PSCellSize =
HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC
End If
```

C++ Example

```
IHESessionGraphics* pGraphics;
pSess->get_Graphics(&pGraphics);
HOSTEX_GRAPHICS_CELL_SIZE CellSize;
pGraphics->get_PSCellSize (&CellSize);
if (CellSize!= HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC)
{
CellSize = HOSTEX_GRAPHICS_CELLSIZE_AUTOMATIC;
pGraphics->put_PSCellSize(CellSize);
}
```

Related Topics

HOSTEX_GRAPHICS_CELLSIZE Data Type

SessionKeyboard Interface

The SessionKeyboard interface lets you set configuration settings related to the keyboard.

Properties

The SessionKeyboard interface consists of the following properties:

AllowAIDKeyRepeat

AllowDiac

CurrentKeyboard

KeyboardType

LockOnAttention

MapNumLock

RemapKeypad

VTCursorKeyApplMode

VTEnableBreak

VTKeypadApplMode

VTNewLineMode

Property: IHESessionKeyboard::AllowAIDKeyRepeat 3270

This property returns or sets a value indicating whether HostExplorer can send multiple function key commands to the host without requiring you to lift and press a key again. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionKeyboard.**AllowAIDKeyRepeat**

SessionKeyboard.**AllowAIDKeyRepeat** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_AllowAIDKeyRepeat**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_AllowAIDKeyRepeat**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer can send multiple function-key commands to the host. A returned value of VARIANT_FALSE indicates that HostExplorer does not send multiple function-key commands to the host.

newVal—A value of VARIANT_TRUE indicates that HostExplorer can send multiple function-key commands to the host. A value of VARIANT_FALSE indicates that HostExplorer does not send multiple function-key commands to the host.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.AllowAIDKeyRepeat

If (bVal = False) Then

'Set value

SessKeyb.AllowAIDKeyRepeat = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_AllowAIDKeyRepeat (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_AllowAIDKeyRepeat (bVal);

}
```

Property: IHESessionKeyboard::AllowDiac 3270 5250

This property returns or sets a value indicating whether HostExplorer supports accented and/or special characters. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionKeyboard.**AllowDiac**

SessionKeyboard.**AllowDiac** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_AllowDiac**([out, retval] VARIANT_BOOL *
pVal);

HRESULT IHESessionKeyboard::**put_AllowDiac**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer supports accented and/or special characters. A returned value of VARIANT_FALSE indicates that HostExplorer does not support accented and/or special characters.

newVal—A value of VARIANT_TRUE indicates that HostExplorer supports accented and/or special characters. A value of VARIANT_FALSE indicates that HostExplorer does not support accented and/or special characters.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.AllowDiac

If (bVal = False) Then

'Set value

SessKeyb.AllowDiac = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_AllowDiac (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_AllowDiac (bVal);

}
```

Property: IHESessionKeyboard::CurrentKeyboard 3270 5250 VT

This property returns or sets a string specifying a keyboard map to use for the current session.

Basic Syntax

String = SessionKeyboard.**CurrentKeyboard**
SessionKeyboard.**CurrentKeyboard** = String

C++ Syntax

HRESULT IHESessionKeyboard::**get_CurrentKeyboard**([out, retval] BSTR * *pVal*);
HRESULT IHESessionKeyboard::**put_CurrentKeyboard**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the keyboard map to use for the current session.
newVal—The set string, specifying the keyboard map to use for the current session.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim strVal As String

'Get string

strVal = SessKeyb.CurrentKeyboard

If (Len(strVal) = 0) Then

'Set string

SessKeyb.CurrentKeyboard = "Default"

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

BSTR bstr ;

pKeyb->get_CurrentKeyboard (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("Default"));

pKeyb->put_CurrentKeyboard (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionKeyboard::KeyboardType 3270 5250 VT

This property returns or sets a value that specifies the type of keyboard to use for the current session.

Basic Syntax

HOSTEX_KEYBOARD_TYPE = SessionKeyboard.**KeyboardType**

SessionKeyboard.**KeyboardType** = HOSTEX_KEYBOARD_TYPE

C++ Syntax

HRESULT IHESessionKeyboard::**get_KeyboardType**([out, retval] HOSTEX_KEYBOARD_TYPE * *pVal*);

HRESULT IHESessionKeyboard::**put_KeyboardType**([in] HOSTEX_KEYBOARD_TYPE *newVal*);

Parameters

pVal—The returned value, specifying the type of keyboard to use for the current session.

newVal—The set value, specifying the type of keyboard to use for the current session.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim kbType As HOSTEX_KEYBOARD_TYPE

kbType = SessKeyb.KeyboardType

If (kbType <> HOSTEX_KEYBOARD_TYPE_PC_101) Then

SessKeyb.KeyboardType =

HOSTEX_KEYBOARD_TYPE_PC_101

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

HOSTEX_KEYBOARD_TYPE kbType;

pKeyb->get_KeyboardType(&kbType);

if (kbType != HOSTEX_KEYBOARD_TYPE_PC_101)

{

kbType = HOSTEX_KEYBOARD_TYPE_PC_101;

pKeyb->put_KeyboardType (kbType);

}
```

Related Topics

HOSTEX_KEYBOARD_TYPE Data Type

Property: IHESessionKeyboard::LockOnAttention 3270 5250 VT

This property returns or sets a value that indicates whether the keyboard is locked after you send an Attention-key command. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionKeyboard.**LockOnAttention**

SessionKeyboard.**LockOnAttention** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_LockOnAttention**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_LockOnAttention**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the keyboard is locked. A returned value of VARIANT_FALSE indicates that the keyboard is not locked.

newVal—A value of VARIANT_TRUE indicates that the keyboard is locked. A value of VARIANT_FALSE indicates that the keyboard is not locked.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.LockOnAttention

If (bVal = False) Then

'Set value

SessKeyb.LockOnAttention = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_LockOnAttention (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_LockOnAttention (bVal);

}
```

Property: IHESessionKeyboard::MapNumLock 3270 5250 VT

This property returns or sets a value indicating whether you can use the numeric keypad, regardless of the NumLock mode. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionKeyboard.**MapNumLock**

SessionKeyboard.**MapNumLock** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_MapNumLock**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_MapNumLock**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that you can use the numeric keypad. A returned value of VARIANT_FALSE indicates that you cannot use the numeric keypad.

newVal—A value of VARIANT_TRUE indicates that you can use the numeric keypad. A value of VARIANT_FALSE indicates that you cannot use the numeric keypad.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.MapNumLock

If (bVal = False) Then

'Set value

SessKeyb.MapNumLock = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_MapNumLock (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_MapNumLock (bVal);

}
```

Property: IHESessionKeyboard::RemapKeypad 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer allows you to map the /, ”, -, and + keys while in NumLock mode. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionKeyboard.**RemapKeypad**

SessionKeyboard.**RemapKeypad** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_RemapKeypad**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_RemapKeypad**([in] VARIANT_BOOL *newVal*);

Parameters

pVal— A returned value of VARIANT_TRUE indicates that HostExplorer allows you to map the /, ”, -, and + keys while in NumLock mode. A returned value of VARIANT_FALSE indicates that HostExplorer does not allow you to map those keys while in NumLock mode.

newVal—A value of VARIANT_TRUE indicates that HostExplorer allows you to map the /, ”, -, and + keys while in NumLock mode. A value of VARIANT_FALSE indicates that HostExplorer does not allow you to map those keys while in NumLock mode.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.RemapKeypad

If (bVal = False) Then

'Set value

SessKeyb.RemapKeypad = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_RemapKeypad (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_RemapKeypad (bVal);

}
```

Property: IHESessionKeyboard::VTCursorKeyApplMode VT

This property returns or sets a variable indicating the cursor-key mode. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionKeyboard.**VTCursorKeyApplMode**

SessionKeyboard.**VTCursorKeyApplMode** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_VTCursorKeyApplMode**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_VTCursorKeyApplMode**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the cursor key is in Application mode. A returned value of VARIANT_FALSE indicates that the cursor key operates in Normal mode.

newVal—A value of VARIANT_TRUE indicates that the cursor key is in Application mode. A value of VARIANT_FALSE indicates that the cursor key operates in Normal mode.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.VTCursorKeyApplMode

If (bVal = False) Then

'Set value

SessKeyb.VTCursorKeyApplMode = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_VTCursorKeyApplMode (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_VTCursorKeyApplMode (bVal);

}
```

Property: IHESessionKeyboard::VTEnableBreak VT

This property returns or sets a variable indicating whether HostExplorer enables the Break key to send a break signal to the host.

Basic Syntax

Boolean = SessionKeyboard.**VTEnableBreak**

SessionKeyboard.**VTEnableBreak** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_VTEnableBreak**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_VTEnableBreak**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables the Break key. A returned value of VARIANT_FALSE indicates that HostExplorer does not enable the Break key.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables the Break key. A value of VARIANT_FALSE indicates that HostExplorer does not enable the Break key.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.VTEnableBreak

If (bVal = False) Then

'Set value

SessKeyb.VTEnableBreak = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_VTEnableBreak (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_VTEnableBreak (bVal);

}
```

Property: IHESessionKeyboard::VTKeypadApplMode VT

This property returns or sets a variable indicating the Keypad mode. When set to VARIANT_TRUE, the keypad operates in Application mode. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionKeyboard.**VTKeypadApplMode**

SessionKeyboard.**VTKeypadApplMode** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_VTKeypadApplMode**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_VTKeypadApplMode**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the keypad operates in Application mode. A returned value of VARIANT_FALSE indicates that the keypad operates in Numeric mode.

newVal—A value of VARIANT_TRUE indicates that the keypad operates in Application mode. A value of VARIANT_FALSE indicates that the keypad operates in Numeric mode.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.VTKeypadApplMode

If (bVal = False) Then

'Set value

SessKeyb.VTKeypadApplMode = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_VTKeypadApplMode (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_VTKeypadApplMode (bVal);

}
```

Property: IHESessionKeyboard::VTNewLineMode VT

This property returns or sets a variable that determines whether pressing Enter sends a carriage return (CR) or carriage return/line feed (CR/LF) to the host. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionKeyboard.**VTNewLineMode**

SessionKeyboard.**VTNewLineMode** = Boolean

C++ Syntax

HRESULT IHESessionKeyboard::**get_VTNewLineMode**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionKeyboard::**put_VTNewLineMode**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that pressing Enter sends a CR to the host. A returned value of VARIANT_FALSE indicates that pressing Enter sends a CR/LF to the host.

newVal—A value of VARIANT_TRUE indicates that pressing Enter sends a CR to the host. A value of VARIANT_FALSE indicates that pressing Enter sends a CR/LF to the host.

Basic Example

```
Dim SessKeyb As HESessionKeyboard

Set SessKeyb = Session.Keyboard

Dim bVal As Boolean

'Get value

bVal = SessKeyb.VTNewLineMode

If (bVal = False) Then

'Set value

SessKeyb.VTNewLineMode = True

End If
```

C++ Example

```
IHESessionKeyboard* pKeyb;

pSess->get_Keyboard(&pKeyb);

VARIANT_BOOL bVal;

pKeyb->get_VTNewLineMode (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pKeyb->put_VTNewLineMode (bVal);

}
```

SessionMouse Interface

The SessionMouse interface lets you set configuration settings related to the mouse.

Properties

The SessionMouse interface consists of the following property:

BlockSelect

Property: IHESessionMouse::BlockSelect VT

This property returns or sets a value indicating whether HostExplorer selects a rectangular region of the screen when selecting text. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionMouse.**BlockSelect**

SessionMouse.**BlockSelect** = Boolean

C++ Syntax

HRESULT IHESessionMouse::**get_BlockSelect**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionMouse::**put_BlockSelect**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indciates that HostExplorer selects a rectangular region of the screen when selecting text. A returned value of VARIANT_FALSE indicates that HostExplorer selects text in a stream-like fashion.

newVal—A value of VARIANT_TRUE indciates that HostExplorer selects a rectangular region of the screen when selecting text. A value of VARIANT_FALSE indicates that HostExplorer selects text in a stream-like fashion.

Basic Example

```
Dim SessMouse As HESessionMouse

Set SessMouse = Session.Mouse

Dim bVal As Boolean

'Get value

bVal = SessMouse.BlockSelect

If (bVal = False) Then

'Set value

SessMouse.BlockSelect = True

End If
```

C++ Example

```
IHESessionMouse* pMouse;

pSess->get_Mouse(&pMouse);

VARIANT_BOOL bVal;

pMouse->get_BlockSelect(&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pMouse->put_BlockSelect(bVal);

}
```

SessionTrackMenu Interface

The SessionTrackMenu interface lets you set configuration settings related to the Track Menu.

Properties

The SessionTrackMenu interface consists of the following properties:

TrackCommands

TrackLabels

Property: IHESessionTrackMenu::TrackCommands 3270 5250 VT

This property returns or sets a string specifying the Track Menu functions, which are represented as predefined strings. To determine these strings, open a HostExplorer session, then open the Edit Session Properties dialog box. In the Track Menu section, the labels correspond to the functions. A function and label usually have the same string value.

Basic Syntax

String = SessionTrackMenu.**TrackCommands**

SessionTrackMenu.**TrackCommands** = String

C++ Syntax

HRESULT IHESessionTrackMenu::**get_TrackCommands**([out, retval] BSTR * *pVal*);

HRESULT IHESessionTrackMenu::**put_TrackCommands**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the Track menu functions.

newVal—The set string, specifying the Track menu functions. These functions must correspond to the labels that you chose in the "SessionTrackMenu.TrackLabels" property.

Basic Example

```
Dim strVal As String

'Get string

strVal = SessTrk.TrackCommands

If (Len(strVal) = 0) Then

SessTrk.TrackCommands =

"Edit-Copy, Edit-Paste,Select-Line"

End If
```

C++ Example

```
IHESessionTrackMenu* pTrack;

pSess->get_TrackMenu(&pTrack);

BSTR bstr ;

pTrack->get_TrackCommands(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR

("Edit-Copy, Edit-Paste,Select-Line"));

pTrack->put_TrackCommands(bstr);

SysFreeString(bstr);

}
```

Property: IHESessionTrackMenu::TrackLabels 3270 5250 VT

This property returns or sets a string specifying the Track menu labels.

Basic Syntax

String = SessionTrackMenu.**TrackLabels**
SessionTrackMenu.**TrackLabels** = String

C++ Syntax

HRESULT IHESessionTrackMenu::**get_TrackLabels**([out, retval] BSTR * *pVal*);
HRESULT IHESessionTrackMenu::**put_TrackLabels**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the Track menu labels.
newVal—The set string, specifying the Track menu labels. These labels must correspond to the commands that you chose in the "SessionTrackMenu.TrackCommands" property.

Basic Example

```
Dim strVal As String

'Get string

strVal = SessTrk.TrackLabels

If (Len(strVal) <> 0) Then

SessTrk.TrackLabels = "Copy,Paste,Select"

End If
```

C++ Example

```
IHESessionTrackMenu* pTrack;

pSess->get_TrackMenu(&pTrack);

BSTR bstr ;

pTrack->get_TrackLabels (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR(""Copy,Paste,

Select "));

pTrack->put_TrackLabels (bstr);

SysFreeString(bstr);

}
```

SessionTranslationTable Interface

The Translation Table interface lets you change the translation table (or host code page) used to display data received from the host. Because mainframe systems and midrange systems (such as the AS/400) support many host languages, you must select the correct translation table to display host data properly.

Properties

The SessionTranslationTable interface consists of the following property:

CurrentLanguage

Property: IHESessionTranslationTable::CurrentLanguage 3270 5250 VT

This property returns or sets a value specifying the host code page.

Basic Syntax SessionTranslationTable.**CurrentLanguage** = String

C++ Syntax HRESULT IHESessionTranslationTable::**get_CurrentLanguage**([out, retval] BSTR * *pVal*);
HRESULT IHESessionTranslationTable::**put_CurrentLanguage**([in] BSTR *newVal*);

Parameters *pVal*—The returned value, specifying the host code page.
newVal—The set value, specifying the host code page.

Basic Example

```
Dim SessTrans As HESessionTranslationTable

Set SessTrans = Session.TranslationTable

Dim strVal As String

strVal = SessTrans.CurrentLanguage

if (Len(strVal)=0) Then

SessTrans.CurrentLanguage=”Default”

End If
```

C++ Example

```
IHESessionTranslationTable* pTrans;

pSess->get_TranslationTable (&pTrans);

BSTR bstr ;

pTrans->get_CurrentLanguage(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR(“Default "));

pTrans->put_CurrentLanguage(bstr);

SysFreeString(bstr);

}
```

SessionVTCharset Interface

The SessionVTCharset interface lets you set configuration settings related to the VT Character Set.

Properties

The SessionVTCharset interface consists of the following properties:

VTNRC

VTNRCMode

VTUPSS

Property: IHESessionVTCharset::VTNRC VT

This property returns or sets the NRC (National Replacement Character) set.

Basic Syntax

Integer = SessionVTCharset.**VTNRC**
SessionVTCharset.**VTNRC** = Integer

C++ Syntax

HRESULT IHESessionVTCharset::**get_VTNRC**([out, retval] short * *pVal*);
HRESULT IHESessionVTCharset::**put_VTNRC**([in] short *newVal*);

Parameters

pVal—The returned NRC set.
newVal—The NRC set that you specify.

Basic Example

```
Dim iVal As Integer

Dim VTCharSet As HESessionVTCharSet

Set VTCharSet = Session.VTCharSet

iVal = VTCharSet.VTNRC



‘ISO French is hex 0x0052, which corresponds

‘to 82 in base ten.

If (iVal = 82) Then

‘DEC Finnish is hex 0x0043, which

‘ corresponds to 67 in base ten

VTCharSet.VTNRC = 67

End If
```

C++ Example

```
IHESessionVTCharSet *pVtChSet;

pSess->get_VTCharSet(&pVtChSet);

short sVal;

pVtChSet->get_VTNRC (&sVal);

//ISO French is hex 0x0052, which

//corresponds to 82 in base ten.

if (sVal==82)

{

//DEC Finnish is hex 0x0043, which

//corresponds to 67 in base ten

sVal = 67;

pVtChSet->put_VTNRC (sVal);

}
```

Related Topics

TNVT NRC Language-Conversion Table

Property: IHESessionVTCharset::VTNRCMode VT

This property returns or sets a value indicating whether NRC (National Replacement Character) 7-bit mode is set for the current VT session.

Basic Syntax

Boolean = SessionVTCharset.**VTNRCMode**

SessionVTCharset.**VTNRCMode** = Boolean

C++ Syntax

HRESULT IHESessionVTCharset::**get_VTNRCMode**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionVTCharset::**put_VTNRCMode**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that NRC 7-bit mode is set for the current session. A returned value of VARIANT_FALSE indicates that NRC 7-bit mode is not set for the current session.

newVal—A value of VARIANT_TRUE indicates that NRC 7-bit mode is set for the current session. A value of VARIANT_FALSE indicates that NRC 7-bit mode is not set for the current session.

Basic Example

```
Dim VTCharSet As HESessionVTCharSet

Set VTCharSet = Session.VTCharSet

Dim bVal As Boolean

bVal = VTCharSet.VTNRCMode

If (bVal = False) Then

VTCharSet.VTNRCMode = True

End If
```

C++ Example

```
IHESessionVTCharSet * pVtChSet;

pSess->get_VTCharSet(&pVtChSet);

VARIANT_BOOL bVal;

pVtChSet->get_VTNRCMode(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pVtChSet->put_VTNRCMode(bVal);

}
```

Related Topics

TNVT NRC Language-Conversion Table

Property: IHESessionVTCharset::VTUPSS VT

This property returns or sets the UPSS (User Preferred Supplemental Character Set).

Basic Syntax

Integer = SessionVTCharset.**VTUPSS**
SessionVTCharset.**VTUPSS** = Integer

C++ Syntax

HRESULT IHESessionVTCharset::**get_VTUPSS**([out, retval] short * *pVal*);
HRESULT IHESessionVTCharset::**put_VTUPSS**([in] short *newVal*);

Parameters

pVal—The returned UPSS in base ten.
newVal—The set UPSS in base ten.

Basic Example

```
Dim iVal As Integer

Dim VTCharSet As HESessionVTCharSet

Set VTCharSet = Session.VTUPSS

iVal = VTCharSet.VTNRC

‘ISO Cyrillic is hex 0x000C, which

‘ corresponds to 12 in base ten.

If (iVal = 12) Then

‘PC Estonian is hex 0x0019, which

‘ corresponds to 25 in base ten

VTCharSet.VTUPSS = 25

End If
```

C++ Example

```
IHESessionVTCharSet *pVtChSet;
pSess->get_VTCharSet(&pVtChSet);
short sVal;
pVtChSet->get_VTUPSS (&sVal);
//PC Estonian is hex 0x0019, which
//corresponds to 25 in base ten
if (sVal==12)
{
//PC Estonian is hex 0x0019, which
//corresponds to 25 in base ten
sVal = 25;
pVtChSet->put_VTUPSS (sVal);
}
```

Related Topics

TNVT UPSS Language-Conversion Table

SessionEditing Interface

The SessionEditing interface lets you set configuration settings related to editing.

Properties

The SessionEditing interface consists of the following properties:

AutoCopy

AutoCopyKeepSelection

BellMargin

CellDelimited

ClipFormatBitmap

ClipFormatCSV

ClipFormatHE

ClipFormatPasteLink

ClipFormatRTF

ClipFormatText

CutChar

CutMode

EntryAssist

LeftMargin

MoveCursorAfterPaste

PasteChar

PasteMode

RightMargin

WordWrap

Property: IHESessionEditing::AutoCopy 3270 5250 VT

This property returns or sets a value that indicates whether HostExplorer automatically copies all selected text to the clipboard. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionEditing.**AutoCopy**

SessionEditing.**AutoCopy** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_AutoCopy**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_AutoCopy**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically copies all selected text to the clipboard. A returned value of VARIANT_FALSE indicates that HostExplorer does not automatically copy all selected text to the clipboard.

newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically copies all selected text to the clipboard. A value of VARIANT_FALSE indicates that HostExplorer does not automatically copy all selected text to the clipboard.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.AutoCopy

If (bVal = False) Then

'Set value

SessEd.AutoCopy = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_AutoCopy(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_AutoCopy(bVal);

}
```

Property: IHESessionEditing::AutoCopyKeepSelection 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer maintains the selection once you have copied or cut text. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionEditing.**AutoCopyKeepSelection**

SessionEditing.**AutoCopyKeepSelection** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_AutoCopyKeepSelection**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_AutoCopyKeepSelection**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer maintains the selection. A returned value of VARIANT_FALSE indicates that HostExplorer does not maintain the selection.

newVal—A value of VARIANT_TRUE indicates that HostExplorer maintains the selection. A value of VARIANT_FALSE indicates that HostExplorer does not maintain the selection.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.AutoCopyKeepSelection

If (bVal = False) Then

'Set value

SessEd.AutoCopyKeepSelection = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_AutoCopyKeepSelection(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_AutoCopyKeepSelection

(bVal);

}
```

Property: IHESessionEditing::BellMargin 3270 5250 VT

This property creates a beeping sound when the cursor reaches the last column of an input field.

Basic Syntax

Integer = SessionEditing.**BellMargin**
SessionEditing.**BellMargin** = Integer

C++ Syntax

HRESULT IHESessionEditing::**get_BellMargin**([out, retval] short * *pVal*);
HRESULT IHESessionEditing::**put_BellMargin**([in] short *newVal*);

Parameters

pVal—The returned value, indicating the last column.
newVal—The set value, indicating the last column.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim iVal As Integer

‘Get value

iVal = SessEd.BellMargin

If (iVal = 0) Then

‘Set value

SessEd.BellMargin = 7

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

short sVal;

pEdit->get_BellMargin(&sVal);

if (sVal==0)

{

sVal = 7;

pEdit->put_BellMargin(sVal);

}
```

Property: IHESessionEditing::CellDelimited 3270 5250 VT

This property enables CSV (Comma Separated Value) and BIF (Binary Interchange File) formats when copying data to the clipboard and pasting data from other applications. When copying data to the clipboard in Cell Delimited format, HostExplorer can parse screen data at words or at field attributes. This lets you determine how data appears in cells in your spreadsheet application.

Basic Syntax

HOSTEX_CELL_DELIMITED = SessionEditing.**CellDelimited**

SessionEditing.**CellDelimited** = HOSTEX_CELL_DELIMITED

C++ Syntax

HRESULT IHESessionEditing::**get_CellDelimited**([out, retval] HOSTEX_CELL_DELIMITED * *pVal*);

HRESULT IHESessionEditing::**put_CellDelimited**([in] HOSTEX_CELL_DELIMITED *newVal*);

Parameters

pVal—The returned value, which indicates how data appears in cells.

newVal—The set value, which indicates how data appears in cells.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim CellDelim As HOSTEX_CELL_DELIMITED

CellDelim = SessEd.CellDelimited

If (CellDelim <>

HOSTEX_CELL_DELIMITED_FIELD) Then

SessEd.CellDelimited =

HOSTEX_CELL_DELIMITED_FIELD

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

HOSTEX_CELL_DELIMITED CellDelim;

pEdit->get_CellDelimited(&CellDelim);

if (CellDelim!= HOSTEX_CELL_DELIMITED_FIELD)

{

CellDelim =

HOSTEX_CELL_DELIMITED_FIELD;

pEdit->put_CellDelimited(CellDelim);

}
```

Related Topics

HOSTEX_CELL_DELIMITED Data Type

Property: IHESessionEditing::ClipFormatBitmap 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables bitmap format when copying data to the clipboard. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**ClipFormatBitmap**

SessionEditing.**ClipFormatBitmap** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_ClipFormatBitmap**([out, retval] VARIANT_BOOL *
pVal);

HRESULT IHESessionEditing::**put_ClipFormatBitmap**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables bitmap format. A returned value of VARIANT_FALSE indicates that HostExplorer disables the Bitmap format.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables bitmap format. A value of VARIANT_FALSE indicates that HostExplorer disables the Bitmap format.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.ClipFormatBitmap

If (bVal = False) Then

'Set value

SessEd.ClipFormatBitmap = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_ClipFormatBitmap (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_ClipFormatBitmap (bVal);

}
```

Property: IHESessionEditing::ClipFormatCSV 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables CSV (Comma Separated Value) and BIF (Binary Interchange File) formats when copying data to the clipboard and pasting data from other applications. CSV and BI are common formats used by spreadsheet applications. When copying data to the clipboard in Cell Delimited format, HostExplorer can parse screen data at words or at field attributes. This lets you determine how data appears in cells in your spreadsheet application. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**ClipFormatCSV**

SessionEditing.**ClipFormatCSV** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_ClipFormatCSV**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_ClipFormatCSV**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables CSV and BIF formats. A returned value of VARIANT_FALSE indicates that HostExplorer disables CSV and BIFF formats.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables CSV and BIF formats. A value of VARIANT_FALSE indicates that HostExplorer disables CSV and BIFF formats.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.ClipFormatCSV

If (bVal = False) Then

'Set value

SessEd.ClipFormatCSV = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_ClipFormatCSV (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_ClipFormatCSV (bVal);

}
```

Property: IHESessionEditing::ClipFormatHE 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables its proprietary format when copying data to the clipboard. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**ClipFormatHE**

SessionEditing.**ClipFormatHE** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_ClipFormatHE**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_ClipFormatHE**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables its proprietary format when copying data to the clipboard. A returned value of VARIANT_FALSE indicates that HostExplorer disables its proprietary format when copying data to the clipboard.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables its proprietary format when copying data to the clipboard. A value of VARIANT_FALSE indicates that HostExplorer disables its proprietary format when copying data to the clipboard.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.ClipFormatHE

If (bVal = False) Then

'Set value

SessEd.ClipFormatHE = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_ClipFormatHE (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_ClipFormatHE (bVal);

}
```

Property: IHESessionEditing::ClipFormatPasteLink 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables Paste Link format when copying data to the clipboard. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**ClipFormatPasteLink**

SessionEditing.**ClipFormatPasteLink** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_ClipFormatPasteLink**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_ClipFormatPasteLink**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables Paste Link format when copying data to the clipboard. A returned value of VARIANT_FALSE indicates that HostExplorer disables Paste Link format.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables the Paste Link format when copying data to the clipboard. A value of VARIANT_FALSE indicates that HostExplorer disables Paste Link format.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.ClipFormatPasteLink

If (bVal = False) Then

'Set value

SessEd.ClipFormatPasteLink = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_ClipFormatPasteLink (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_ClipFormatPasteLink (bVal);

}
```

Property: IHESessionEditing::ClipFormatRTF 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables RTF (Rich Text Format) when copying data to the clipboard. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**ClipFormatRTF**

SessionEditing.**ClipFormatRTF** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_ClipFormatRTF**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_ClipFormatRTF**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables RTF when copying data to the clipboard. A returned value of VARIANT_FALSE indicates that HostExplorer disables RTF.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables RTF when copying data to the clipboard. A value of VARIANT_FALSE indicates that HostExplorer disables RTF.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.ClipFormatRTF

If (bVal = False) Then

'Set value

SessEd.ClipFormatRTF = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_ClipFormatRTF(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_ClipFormatRTF(bVal);

}
```

Property: IHESessionEditing::ClipFormatText 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables standard text format for clipboard use. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**ClipFormatText**

SessionEditing.**ClipFormatText** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_ClipFormatText**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_ClipFormatText**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables standard text format for clipboard use. A returned value of VARIANT_FALSE indicates that HostExplorer disables standard text format for clipboard use.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables standard text format for clipboard use. A value of VARIANT_FALSE indicates that HostExplorer disables standard text format for clipboard use.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.ClipFormatText

If (bVal = False) Then

'Set value

SessEd.ClipFormatText = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_ClipFormatText (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_ClipFormatText (bVal);

}
```

Property: IHESessionEditing::CutChar 3270 5250

This property returns or sets a value indicating how HostExplorer replaces the field-attribute character when you use the Cut application.

Basic Syntax

Integer = SessionEditing.**CutChar**

SessionEditing.**CutChar** = Integer

C++ Syntax

HRESULT IHESessionEditing::**get_CutChar**([out, retval] short * *pVal*);

HRESULT IHESessionEditing::**put_CutChar**([in] short *newVal*);

Parameters

pVal—One of the following returned values, which indicate how HostExplorer replaces the field-attribute character:

· 0—None

· 1—Tab

· 2—Comma

· 3—Paragraph

newVal—The set value, which indicates how HostExplorer replaces the field-attribute character.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim iVal As Integer

iVal = SessEd.CutChar

If (iVal = 0) Then

SessEd.CutChar = 2

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

short sVal;

pEdit->get_CutChar (&sVal);

if (sVal==0)

{

sVal = 2;

pEdit->put_CutChar(sVal);

}
```

Property: IHESessionEditing::CutMode 3270 5250 VT

This property returns or sets a value indicating how HostExplorer removes selected text from unprotected areas of the screen.

Basic Syntax

Integer = SessionEditing.**CutMode**
SessionEditing.**CutMode** = Integer

C++ Syntax

HRESULT IHESessionEditing::**get_CutMode**([out, retval] short * *pVal*);
HRESULT IHESessionEditing::**put_CutMode**([in] short *newVal*);

Parameters

pVal—The following returned values specify how the selected text is removed:

· 0—Replace with Spaces

· 1—Replace with Nulls

· 2—Delete Text

newVal—The set value, which indicates how the selected text is removed.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim iVal As Integer

iVal = SessEd.CutMode

If (iVal = 0) Then

SessEd.CutMode = 2

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

short sVal;

pEdit->get_CutMode(&sVal);

if (sVal==0)

{

sVal = 2;

pEdit->put_CutMode(sVal);

}
```

Property: IHESessionEditing::EntryAssist 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables Entry Assist. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionEditing.**EntryAssist**

SessionEditing.**EntryAssist** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_EntryAssist**([out, retval] VARIANT_BOOL *
pVal);

HRESULT IHESessionEditing::**put_EntryAssist**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables Entry Assist. A returned value of VARIANT_FALSE indicates that HostExplorer disables Entry Assist.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables Entry Assist. A value of VARIANT_FALSE indicates that HostExplorer disables Entry Assist.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.EntryAssist

If (bVal = False) Then

'Set value

SessEd.EntryAssist = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_EntryAssist (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_EntryAssist (bVal);

}
```

Property: IHESessionEditing::LeftMargin 3270 5250

This property returns or sets a value specifying the left margin of the screen.

Basic Syntax

```
Integer = SessionEditing.LeftMargin
SessionEditing.LeftMargin = Integer
```

C++ Syntax

```
HRESULT IHESessionEditing::get_LeftMargin([out, retval] short * pVal);
HRESULT IHESessionEditing::put_LeftMargin([in] short newVal);
```

Parameters

pVal—The returned value, specifying the left margin of the screen.
newVal—The set value, specifying the left margin of the screen.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim iVal As Integer

'Get value

iVal = SessEd.LeftMargin

If (iVal = 0) Then

'Set value

SessEd.LeftMargin = 7

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

short sVal;

pEdit->get_LeftMargin (&sVal);

if (sVal==0)

{

sVal = 2;

pEdit->put_LeftMargin (sVal);

}
```

Property: IHESessionEditing::MoveCursorAfterPaste 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer automatically repositions the cursor after pasting text. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**MoveCursorAfterPaste**

SessionEditing.**MoveCursorAfterPaste** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_MoveCursorAfterPaste**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_MoveCursorAfterPaste**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically repositions the cursor after pasting text. A returned value of VARIANT_FALSE indicates that HostExplorer does not automatically reposition the cursor after pasting text.

newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically repositions the cursor after pasting text. A value of VARIANT_FALSE indicates that HostExplorer does not automatically reposition the cursor after pasting text.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.MoveCursorAfterPaste

If (bVal = False) Then

'Set value

SessEd.MoveCursorAfterPaste = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_MoveCursorAfterPaste(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_MoveCursorAfterPaste (bVal);

}
```

Property: IHESessionEditing::PasteChar 3270 5250

This property returns or sets a value indicating how HostExplorer replaces the field-attribute character when you use the Paste application.

Basic Syntax

Integer = SessionEditing.**PasteChar**

SessionEditing.**PasteChar** = Integer

C++ Syntax

HRESULT IHESessionEditing::**get_PasteChar**([out, retval] short * *pVal*);

HRESULT IHESessionEditing::**put_PasteChar**([in] short *newVal*);

Parameters

pVal—The following returned values specify how HostExplorer replaces the field-attribute character:

· 0—None

· 1—Tab

· 2—Comma

· 3—Paragraph

newVal—The set value, which indicates how HostExplorer replaces the field-attribute character.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim iVal As Integer

iVal = SessEd.PasteChar

If (iVal = 0) Then

SessEd.PasteChar = 2

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

short sVal;

pEdit->get_PasteChar (&sVal);

if (sVal==0)

{

sVal = 2;

pEdit->put_PasteChar (sVal);

}
```

Property: IHESessionEditing::PasteMode 3270 5250

This property lets you determine how HostExplorer pastes the contents of the clipboard to the current cursor location.

Basic Syntax

Integer = SessionEditing.**PasteMode**

SessionEditing.**PasteMode** = Integer

C++ Syntax

HRESULT IHESessionEditing::**get_PasteMode**([out, retval] short * *pVal*);

HRESULT IHESessionEditing::**put_PasteMode**([in] short *newVal*);

Parameters

pVal—The following returned values specify how HostExplorer pastes the contents of the clipboard to the current cursor location:

· 0—Replace with Spaces

· 1—Replace with Nulls

· 2—Delete Text

newVal—The set string, which indicates how HostExplorer pastes the contents of the clipboard to the current cursor location.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim iVal As Integer

iVal = SessEd.PasteMode

If (iVal = 0) Then

SessEd.PasteMode = 2

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

short sVal;

pEdit->get_PasteMode (&sVal);

if (sVal==0)

{

sVal = 2;

pEdit->put_PasteMode (sVal);

}
```

Property: IHESessionEditing::RightMargin 3270 5250

This property returns or sets a value specifying the right margin of the screen.

Basic Syntax

Integer = SessionEditing.**RightMargin**

SessionEditing.**RightMargin** = Integer

C++ Syntax

HRESULT IHESessionEditing::**get_RightMargin**([out, retval] short * *pVal*);

HRESULT IHESessionEditing::**put_RightMargin**([in] short *newVal*);

Parameters

pVal—The returned value specifying the right margin of the screen.

newVal—The set value specifying the right margin of the screen.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim iVal As Integer

‘Get value

iVal = SessEd.RightMargin

If (iVal = 0) Then

‘Set value

SessEd.RightMargin = 7

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

short sVal;

pEdit->get_RightMargin (&sVal);

if (sVal==0)

{

sVal = 2;

pEdit->put_RightMargin (sVal);

}
```

Property: IHESessionEditing::WordWrap 3270 5250

This property returns or sets a value indicating whether HostExplorer cuts text upon reaching the end of a field or wraps text to the next available field. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionEditing.**WordWrap**

SessionEditing.**WordWrap** = Boolean

C++ Syntax

HRESULT IHESessionEditing::**get_WordWrap**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionEditing::**put_WordWrap**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer wraps text to the next available field. A returned value of VARIANT_FALSE indicates that HostExplorer cuts text when it reaches the end of a field.

newVal—A value of VARIANT_TRUE indicates that HostExplorer wraps text to the next available field. A value of VARIANT_FALSE indicates that HostExplorer cuts text when it reaches the end of a field.

Basic Example

```
Dim SessEd As HESessionEditing

Set SessEd = Session.Editing

Dim bVal As Boolean

'Get value

bVal = SessEd.WordWrap

If (bVal = False) Then

'Set value

SessEd. WordWrap = True

End If
```

C++ Example

```
IHESessionEditing *pEdit;

pSess->get_Editing(&pEdit);

VARIANT_BOOL bVal;

pEdit->get_WordWrap (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pEdit->put_WordWrap(bVal);

}
```

SessionFileTransfer Interface

The SessionFileTransfer interface lets you set configuration settings related to transferring files.

Properties

The SessionFileTransfer interface consists of the following properties:

Append
ASCII
AutoCC
AutoClearMonitor
BlkSize
CRLF
DefaultDownloadPath
DefaultProtocol
DefaultRecvDir
DefaultUploadPath
DownloadHostFileName
DownloadPCFileName
ExtraOptions
FileExistAction
Host
INDFileName
ShowRecvDir
UploadHostFileName
UploadPCFileName
UserDefinedDownload
UserDefinedUpload
XferBlockSize
XferDest
XferHostCodePage
XferPCCodePage
XferSource
XferStartAction
Xm1KPacket
XmAckTimeout
XmCRC
YmAckTimeout
YmUseFullPath

KmBinaryPrefix
KmRLE
KmTextMode
KmUseFullPath
Lrecl
QuickMode
Recfm
ZmAutoDownload
ZmCrashRecovery
ZmMaxErrors
ZmOverwriteMngmt
ZmSlideWindow
ZmUseFullPath
ZmWindowSize

Property: IHESessionFileTransfer::Append 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer appends a file to an existing file on the host.

Basic Syntax

Boolean = SessionFileTransfer.**Append**

SessionFileTransfer.**Append** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_Append**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_Append**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer appends the file. A returned value of VARIANT_FALSE indicates that HostExplorer does not append the file.

newVal—A value of VARIANT_TRUE indicates that HostExplorer appends the file. A value of VARIANT_FALSE indicates that HostExplorer does not append the file.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.Append

If (bVal = False) Then

'Set value

SessXfr.Append = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_Append(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_Append(bVal);

}
```

Property: IHESessionFileTransfer::ASCII 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer translates an ASCII (a character set used on a personal computer) file to an EBCDIC (an IBM host character set) file.

Basic Syntax

Boolean = SessionFileTransfer.**Ascii**

SessionFileTransfer.**Ascii** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_Ascii**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_Ascii**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer translates an ASCII file to an EBCDIC file. A returned value of VARIANT_FALSE indicates that HostExplorer does not translate an ASCII file to an EBCDIC file.

newVal—A value of VARIANT_TRUE indicates that HostExplorer translates an ASCII file to an EBCDIC file. A value of VARIANT_FALSE indicates that HostExplorer does not translate an ASCII file to an EBCDIC file.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.Ascii

If (bVal = False) Then

'Set value

SessXfr.Ascii = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_Ascii (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_Ascii (bVal);

}
```

Property: IHESessionFileTransfer::AutoCC 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer converts carriage control on the host to carriage control on your computer. This property applies only when you download files from the host.

Basic Syntax

Boolean = SessionFileTransfer.**AutoCC**

SessionFileTransfer.**AutoCC** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_AutoCC**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_AutoCC**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer converts carriage control on the host to carriage control on your computer. A returned value of VARIANT_FALSE indicates that HostExplorer does not convert carriage control on the host to carriage control on your computer.

newVal—A value of VARIANT_TRUE indicates that HostExplorer converts carriage control on the host to carriage control on your computer. A value of VARIANT_FALSE indicates that HostExplorer does not convert carriage control on the host to carriage control on your computer.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.AutoCC

If (bVal = False) Then

'Set value

SessXfr.AutoCC = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_AutoCC (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_AutoCC (bVal);

}
```

Property: IHESessionFileTransfer::AutoClearMonitor 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer automatically exits the File Transfer Monitor when it has finished transferring a file.

Basic Syntax

Boolean = SessionFileTransfer.**AutoClearMonitor**

SessionFileTransfer.**AutoClearMonitor** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_AutoClearMonitor**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_AutoClearMonitor**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically exits the File Transfer Monitor. A returned value of VARIANT_FALSE indicates that HostExplorer does not automatically exit the File Transfer Monitor.

newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically exits the File Transfer Monitor. A value of VARIANT_FALSE indicates that HostExplorer does not automatically exit the File Transfer Monitor.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.AutoClearMonitor

If (bVal = False) Then

'Set value

SessXfr.AutoClearMonitor = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_AutoClearMonitor (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_AutoClearMonitor (bVal);

}
```

Property: IHESessionFileTransfer::BlkSize 3270

This property returns or sets a value indicating the block size that HostExplorer uses when it transfers files. The size of the block ranges from 256 bytes to 32768 bytes. By default, this property is set to 2048 bytes.

Basic Syntax

Long = SessionFileTransfer.**BlkSize**
SessionFileTransfer.**BlkSize** = Long

C++ Syntax

HRESULT IHESessionFileTransfer::**get_BlkSize**([out, retval] long * *pVal*);
HRESULT IHESessionFileTransfer::**put_BlkSize**([in] long *newVal*);

Parameters

pVal—The returned value, which indicates the block size of a file.
newVal—The set value, which indicates the block size of a file.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim lVal As Long

'Get value

lVal = SessXfr.BlkSize

If (lVal = 256) Then

'Set value

SessXfr.BlkSize = 1024

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

long lVal;

pXfr->get_BlkSize(&lVal);

if (lVal==256)

{

lVal = 1024;

pXfr->put_BlkSize(lVal);

}
```

Property: IHESessionFileTransfer::CRLF 3270

This property returns or sets a value indicating whether HostExplorer translates CR/LF (carriage return/line feed) characters to records on the host file system. This property is normally required when you transfer text files.

Basic Syntax

Boolean = SessionFileTransfer.**CRLF**
SessionFileTransfer.**CRLF** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_CRLF**([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHESessionFileTransfer::**put_CRLF**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer translates CR/LF characters. A returned value of VARIANT_FALSE indicates that HostExplorer does not translate CR/LF characters.
newVal—A value of VARIANT_TRUE indicates that HostExplorer translates CR/LF characters. A value of VARIANT_FALSE indicates that HostExplorer does not translate CR/LF characters.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.CRLF

If (bVal = False) Then

'Set value

SessXfr.CRLF = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_CRLF (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_CRLF (bVal);

}
```

Property: IHESessionFileTransfer::DefaultDownloadPath 3270 5250 VT

This property returns or sets a string specifying the default directory for downloaded files.

Basic Syntax

String = SessionFileTransfer.**DefaultDownloadPath**
SessionFileTransfer.**DefaultDownloadPath** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_DefaultDownloadPath**([out, retval] BSTR * *pVal*);
HRESULT IHESessionFileTransfer::**put_DefaultDownloadPath**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the default download directory.
newVal—The set string, specifying the default download directory.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.DefaultDownloadPath

If (Len(strVal) = 0) Then

SessXfr.DefaultDownloadPath

= "C:\"

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_DefaultDownloadPath (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\"));

pXfr->put_DefaultDownloadPath (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::DefaultProtocol VT

This property returns or sets a value indicating the default file-transfer protocol.

Basic Syntax

Integer = SessionFileTransfer.**DefaultProtocol**
SessionFileTransfer.**DefaultProtocol** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_DefaultProtocol**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_DefaultProtocol**([in] short *newVal*);

Parameters

pVal—The following returned values indicate the default file-transfer protocol:

· 0—Xmodem

· 1—Ymodem

· 2—Kermit

· 3—Zmodem

newVal—The set value, which indicates the default file-transfer protocol.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.DefaultProtocol

If (iVal=2) Then

'Set value

SessXfr.DefaultProtocol = 3

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_DefaultProtocol(&sVal);

if (sVal==2)

{

sVal = 3;

pXfr->put_DefaultProtocol(sVal);

}
```

Property: IHESessionFileTransfer::DefaultRecvDir 3270 5250 VT

This property returns or sets a string indicating the default directory for received files.

Basic Syntax

String = SessionFileTransfer.**DefaultRecvDir**
SessionFileTransfer.**DefaultRecvDir** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_DefaultRecvDir**([out, retval] BSTR * *pVal*);
HRESULT IHESessionFileTransfer::**put_DefaultRecvDir**([in] BSTR *newVal*);

Parameters

pVal—The returned string, which indicates the default receive directory.
newVal—The set string, which indicates the default receive directory.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.DefaultRecvDir

If (Len(strVal) = 0) Then

SessXfr.DefaultRecvDir

= "C:\"

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_DefaultRecvDir (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\"));
```

```
pXfr->put_DefaultRecvDir (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::DefaultUploadPath 3270 5250 VT

This property returns or sets a string specifying the default directory for uploaded files.

Basic Syntax

String = SessionFileTransfer.**DefaultUploadPath**
SessionFileTransfer.**DefaultUploadPath** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_DefaultUploadPath**([out, retval] BSTR * *pVal*);
HRESULT IHESessionFileTransfer::**put_DefaultUploadPath**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the default upload directory.
newVal—The set string, specifying the default upload directory.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.DefaultUploadPath

If (Len(strVal) = 0) Then

SessXfr.DefaultUploadPath = "C:\"

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_DefaultUploadPath (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\"));

pXfr->put_DefaultUploadPath (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::DownloadHostFileName 3270 5250 VT

This property returns or sets a string specifying the name of a host file to download to your computer.

Basic Syntax

String = SessionFileTransfer.**DownloadHostFileName**
SessionFileTransfer.**DownloadHostFileName** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_DownloadHostFileName**([out, retval] BSTR * *pVal*);
HRESULT IHESessionFileTransfer::**put_DownloadHostFileName**([in] BSTR *newVal*);

Parameters

pVal—The returned string specifying the name of a host file to download.
newVal—The set string specifying the name of a host file to download.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.DownloadHostFileName

If (Len(strVal) = 0) Then

SessXfr.DownloadHostFileName

= "C:\a.txt"

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_DownloadHostFileName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\a.txt"));

pXfr->put_DownloadHostFileName (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::DownloadPCFileName 3270 5250 VT

This property returns or sets a string specifying the file name to which HostExplorer saves a downloaded file on your computer.

Basic Syntax

String = SessionFileTransfer.**DownloadPCFileName**

SessionFileTransfer.**DownloadPCFileName** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_DownloadPCFileName**([out, retval] BSTR * *pVal*);

HRESULT IHESessionFileTransfer::**put_DownloadPCFileName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the file name to which HostExplorer saves a downloaded file.

newVal—The set string, specifying the file name to which HostExplorer saves a downloaded file.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.DownloadPCFileName

If (Len(strVal) = 0) Then

SessXfr.DownloadPCFileName

= "C:\b.txt"

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_DownloadPCFileName(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("C:\b.txt"));

pXfr->put_DownloadPCFileName(bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::ExtraOptions 3270 5250 VT

This property returns or sets a string specifying options that are specific to the operating system. You must specify these custom options in the appropriate format.

Basic Syntax

String = SessionFileTransfer.**ExtraOptions**
SessionFileTransfer.**ExtraOptions** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ExtraOptions**([out, retval] BSTR * *pVal*);
HRESULT IHESessionFileTransfer::**put_ExtraOptions**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying options that are specific to the operating system.
newVal—The set string, specifying options that are specific to the operating system.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.ExtraOptions

If (Len(strVal) = 0) Then

‘APND is a parameter that the host

‘application is customized to process

SessXfr.ExtraOptions

= “APND”

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_ExtraOptions (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr = SysAllocString(OLESTR("APND"));

pXfr->put_ExtraOptions (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::FileExistAction 3270 5250 VT

This property returns or sets a string specifying what action HostExplorer performs if the name of a downloaded file already exists in the destination directory.

Basic Syntax

Integer = SessionFileTransfer.**FileExistAction**

SessionFileTransfer.**FileExistAction** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_FileExistAction**([out, retval] short * *pVal*);

HRESULT IHESessionFileTransfer::**put_FileExistAction**([in] short *newVal*);

Parameters

pVal—One of the following returned values, which indicates the action to perform if the name of a downloaded file already exists in the destination directory:

· 0—Overwrite

· 1—Rename

· 2—Skip

newVal—The set value, which indicates the action to perform if the name of a downloaded file already exists in the destination directory.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.FileExistAction

If (iVal=2) Then

'Set value

SessXfr.FileExistAction = 0

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_FileExistAction (&sVal);

if (sVal==2)

{

sVal = 0;

pXfr->put_FileExistAction (sVal);

}
```

Property: IHESessionFileTransfer::Host 3270

This property returns or sets a value indicating the operating system run by the host.

Basic Syntax

Integer = SessionFileTransfer.**Host**
SessionFileTransfer.**Host** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_Host**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_Host**([in] short *newVal*);

Parameters

pVal—One of the following returned values, indicating the address of the current host:

· 257—CMS

· 258—TSO/MUSIC

· 259—CICS

newVal—The address that you set for the current host.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.Host

If (iVal=257) Then

'Set value

SessXfr.Host = 259

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_Host (&sVal);

if (sVal==257)

{

sVal = 259;

pXfr->put_Host (sVal);

}
```

Property: IHESessionFileTransfer::INDFILEName 3270 5250 VT

This property returns or sets a string specifying the name of the file-transfer program to use when uploading and/or downloading files.

Basic Syntax

String = SessionFileTransfer.**INDFILEName**
SessionFileTransfer.**INDFILEName** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_INDFILEName**([out, retval] BSTR * *pVal*);
HRESULT IHESessionFileTransfer::**put_INDFILEName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the name of the file-transfer program.
newVal—The set string, specifying the name of the file-transfer program.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim str As String

'Get value

str = SessXfr.INDFILEName

If (Len(strVal) = 0) Then

'Set value

SessXfr.INDFILEName = “IND$FILE”

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_INDFILEName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr=SysAllocString(OLESTR("IND$FILE"));

pXfr->put_INDFILEName (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::KmBinaryPrefix VT

This property returns or sets a value indicating whether HostExplorer attempts to send 8-bit data characters over a 7-bit channel by prefixing non-printable characters (that is, using a binary prefix).

Basic Syntax

Boolean = SessionFileTransfer.**KmBinaryPrefix**

SessionFileTransfer.**KmBinaryPrefix** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_KmBinaryPrefix**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_KmBinaryPrefix**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer attempts to send 8-bit data characters over a 7-bit channel. A returned value of VARIANT_FALSE indicates that HostExplorer does not attempt to send 8-bit characters over a 7-bit channel.

newVal—A value of VARIANT_TRUE indicates that HostExplorer attempts to send 8-bit data characters over a 7-bit channel. A value of VARIANT_FALSE indicates that HostExplorer does not attempt to send 8-bit characters over a 7-bit channel.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.KmBinaryPrefix

If (bVal = False) Then

'Set value

SessXfr.KmBinaryPrefix= True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_KmBinaryPrefix (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_KmBinaryPrefix (bVal);

}
```

Property: IHESessionFileTransfer::KmRLE 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer uses run-length limited encoding (RLL) to compress data, thereby transferring files more efficiently. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionFileTransfer.**KmRLE**

SessionFileTransfer.**KmRLE** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_KmRLE**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_KmRLE**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer uses RLL. A returned value of VARIANT_FALSE indicates that HostExplorer does not use RLL.

newVal—A value of VARIANT_TRUE indicates that HostExplorer uses RLL. A value of VARIANT_FALSE indicates that HostExplorer does not use RLL.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.KmRLE

If (bVal = False) Then

'Set value

SessXfr.KmRLE = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_KmRLE (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_KmRLE (bVal);

}
```

Property: IHESessionFileTransfer::KmTextMode VT

This property returns or sets a value indicating whether HostExplorer strips the upper bit of each byte as it is received, thus preventing any non-ASCII characters from being saved.

Basic Syntax

Boolean = SessionFileTransfer.**KmTextMode**

SessionFileTransfer.**KmTextMode** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_KmTextMode**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_KmTextMode**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer strips the upper bit of each byte and saves only ASCII characters. A returned value of VARIANT_FALSE indicates that HostExplorer does not strip the upper bit of each byte and saves non-ASCII characters.

newVal—A value of VARIANT_TRUE indicates that HostExplorer strips the upper bit of each byte and saves only ASCII characters. A value of VARIANT_FALSE indicates that HostExplorer does not strip the upper bit of each byte and saves non-ASCII characters.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.KmTextMode

If (bVal = False) Then

'Set value

SessXfr.KmTextMode = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_KmTextMode (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_KmTextMode (bVal);

}
```

Property: IHESessionFileTransfer::KmUseFullPath VT

This property returns or sets a value indicating whether HostExplorer saves a file to the current UNIX host directory. This property also lets you keep the entire path name as the file name.

Basic Syntax

Boolean = SessionFileTransfer.**KmUseFullPath**

SessionFileTransfer.**KmUseFullPath** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_KmUseFullPath**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_KmUseFullPath**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A returned value of VARIANT_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory, and does not keep the entire pathname as the file name.

newVal—A value of VARIANT_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A value of VARIANT_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory, and does not keep the entire pathname as the file name.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.KmUseFullPath

If (bVal = False) Then

'Set value

SessXfr.KmUseFullPath = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_KmUseFullPath (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_KmUseFullPath (bVal);

}
```

Property: IHESessionFileTransfer::Lrecl 3270 5250 VT

This property returns or sets a value indicating the logical record size of a file that you send to the host.

Basic Syntax

```
Long = SessionFileTransfer.Lrecl
SessionFileTransfer.Lrecl = Long
```

C++ Syntax

HRESULT IHESessionFileTransfer::**get_Lrecl**([out, retval] long * *pVal*);
HRESULT IHESessionFileTransfer::**put_Lrecl**([in] long *newVal*);

Parameters

pVal—The returned value, which indicates the logical record size of a file.
newVal—The set value, indicating the logical record size of a file.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.Lrecl

If (iVal=0) Then

'Set value

SessXfr.Lrecl = 5

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_Lrecl (&sVal);

if (sVal==0)

{

sVal = 5;

pXfr->put_Lrecl (sVal);

}
```

Property: IHESessionFileTransfer::QuickMode 3270 5250 VT

This property returns or sets a value indicating the file-transfer mode (text, binary, or custom).

Basic Syntax

Integer = SessionFileTransfer.**QuickMode**
SessionFileTransfer.**QuickMode** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_QuickMode**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_QuickMode**([in] short *newVal*);

Parameters

pVal—The following returned values indicate the file-transfer mode:

· 0—Text

· 1—Binary

· 2—Custom

newVal—The set value, which indicates the file-transfer mode.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.QuickMode

If (iVal=0) Then

'Set value

SessXfr.QuickMode = 2

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_QuickMode (&sVal);

if (sVal==0)

{

sVal = 2;

pXfr->put_QuickMode (sVal);

}
```

Property: IHESessionFileTransfer::Recfm 3270

This property returns or sets a value indicating the type of record format (default, fixed, or variable) to use when you transfer a file.

Basic Syntax

Integer = SessionFileTransfer.**Recfm**
SessionFileTransfer.**Recfm** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_Recfm**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_Recfm**([in] short *newVal*);

Parameters

pVal—The following returned values indicate the type of record format:

· 253—Default

· 254—Fixed

· 255—Variable

· 256—Undefined

newVal—The set value, indicating the type of record format.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.Recfm

If (iVal=256) Then

'Set value

SessXfr. Recfm = 253

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_Recfm (&sVal);

if (sVal==256)

{

sVal = 253;

pXfr->put_Recfm (sVal);

}
```

Property: IHESessionFileTransfer::ShowRecvDir 3270 5250 VT

This property returns or sets a value indicating whether the Receive File dialog box opens each time you receive a file.

Basic Syntax

Boolean = SessionFileTransfer.**ShowRecvDir**

SessionFileTransfer.**ShowRecvDir** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ShowRecvDir**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_ShowRecvDir**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Receive File dialog box opens each time you receive a file. A returned value of VARIANT_FALSE indicates that a file begins to be transferred as soon as you select Receive File From Host from the menu. In this case, for the transfer to complete properly, you must have previously selected a default receive directory and a default protocol.

newVal—A value of VARIANT_TRUE indicates that the Receive File dialog box opens each time you receive a file. A value of VARIANT_FALSE indicates that a file begins to be transferred as soon as you select Receive File From Host from the menu.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.ShowRecvDir

If (bVal = False) Then

'Set value

SessXfr.ShowRecvDir = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_ShowRecvDir (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_ShowRecvDir (bVal);

}
```

Property: IHESessionFileTransfer::UploadHostFileName 3270 5250 VT

This property returns or sets a string specifying the name HostExplorer applies to a file uploaded on the host.

Basic Syntax

String = SessionFileTransfer.**UploadHostFileName**
SessionFileTransfer.**UploadHostFileName** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_UploadHostFileName**([out, retval] BSTR * *pVal*);
HRESULT IHESessionFileTransfer::**put_UploadHostFileName**([in] BSTR *newVal*);

Parameters

pVal—The returned string specifying the name HostExplorer applies to a file uploaded on the host.
newVal—The set string specifying the name HostExplorer applies to a file uploaded on the host.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.UploadHostFileName

If (Len(strVal) = 0) Then

SessXfr.UploadHostFileName

= "C:\b.txt"

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_UploadHostFileName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr=SysAllocString(OLESTR("C:\b.txt"));

pXfr->put_UploadHostFileName (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::UploadPCFileName 3270 5250 VT

This property returns or sets a string specifying the name HostExplorer applies to a file uploaded on your computer.

Basic Syntax

String = SessionFileTransfer.**UploadPCFileName**

SessionFileTransfer.**UploadPCFileName** = String

C++ Syntax

HRESULT IHESessionFileTransfer::**get_UploadPCFileName**([out, retval] BSTR * *pVal*);

HRESULT IHESessionFileTransfer::**put_UploadPCFileName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the name HostExplorer applies to a file uploaded on your computer.

newVal—The set string, specifying the name HostExplorer applies to a file uploaded on your computer.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim strVal As String

strVal = SessXfr.UploadPCFileName

If (Len(strVal) = 0) Then

SessXfr.UploadPCFileName

= "C:\b.txt"

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

BSTR bstr;

pXfr->get_UploadPCFileName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr=SysAllocString(OLESTR("C:\b.txt"));

pXfr->put_UploadPCFileName (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionFileTransfer::UserDefinedDownload 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer performs a user-defined download or a default download.

Basic Syntax

Boolean = SessionFileTransfer.**UserDefinedDownload**

SessionFileTransfer.**UserDefinedDownload** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_UserDefinedDownload**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_UserDefinedDownload**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates a user-defined download. A returned value of VARIANT_FALSE indicates a default download.

newVal—A value of VARIANT_TRUE indicates a user-defined download. A value of VARIANT_FALSE indicates a default download.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.UserDefinedDownload

If (bVal = False) Then

'Set value

SessXfr.UserDefinedDownload = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_UserDefinedDownload (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_UserDefinedDownload (bVal);

}
```

Property: IHESessionFileTransfer::UserDefinedUpload 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer performs a user-defined upload or a default upload.

Basic Syntax

Boolean = SessionFileTransfer.**UserDefinedUpload**

SessionFileTransfer.**UserDefinedUpload** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_UserDefinedUpload**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_UserDefinedUpload**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates a user-defined upload. A returned value of VARIANT_FALSE indicates a default upload.

newVal—A value of VARIANT_TRUE indicates a user-defined upload. A value of VARIANT_FALSE indicates a default upload.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.UserDefinedUpload

If (bVal = False) Then

'Set value

SessXfr.UserDefinedUpload = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_UserDefinedUpload (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_UserDefinedUpload (bVal);

}
```

Property: IHESessionFileTransfer::XferBlockSize 3270 5250 VT

This property returns or sets a value specifying the block size that HostExplorer uses when you transfer files. You can select a block size between 256 and 32768 bytes. By default, this property is set to 2048.

Basic Syntax

Integer = SessionFileTransfer.**XferBlockSize**
SessionFileTransfer.**XferBlockSize** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XferBlockSize**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_XferBlockSize**([in] short *newVal*);

Parameters

pVal—The returned value, specifying the block size that HostExplorer uses when you transfer files.
newVal—The set value, specifying the block size that HostExplorer uses when you transfer files.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.XferBlockSize

If (iVal = 256) Then

'Set value

SessXfr.XferBlockSize = 1024

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_XferBlockSize(&sVal);

if (sVal==256)

{

sVal = 1024;

pXfr->put_XferBlockSize (sVal);

}
```

Property: IHESessionFileTransfer::XferDest 3270

This property returns or sets a value indicating whether a file is being transferred to a file system or the clipboard.

Basic Syntax

Integer = SessionFileTransfer.**XferDest**
SessionFileTransfer.**XferDest** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XferDest**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_XferDest**([in] short *newVal*);

Parameters

pVal—The following returned values specify where a file is being transferred:

· 430—File system

· 431—Clipboard

newVal—The set value, specifying where a file is being transferred.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.XferDest

If (iVal=430) Then

'Set value

SessXfr.XferDest = 431

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_XferDest (&sVal);

if (sVal==430)

{

sVal = 431;

pXfr->put_XferDest (sVal);

}
```

Property: IHESessionFileTransfer::XferHostCodePage 3270

This property returns or sets a value indicating the code page for the host data file.

Basic Syntax

Integer = SessionFileTransfer.**XferHostCodePage**
SessionFileTransfer.**XferHostCodePage** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XferHostCodePage**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_XferHostCodePage**([in] short *newVal*);

Parameters

pVal—The following returned values indicate the code page for the host data file:

· 0—Austrian (273)

· 1—Belgian (037)

· 2—Brazilian (037)

· 3—Canadian French (037)

· 4—Danish (277)

· 5—English UK (285)

· 6—English US (1047)

· 7—English US (037)

· 8—Finnish (278)

· 9—French (297)

· 10—German (273)

· 11—Italian (280)

· 12—Multinational (500)

· 13—Dutch (037)

· 14—Norwegian (277)

· 15—Portuguese (037)

· 16—Spanish (Latin America)

· 17—Swedish (278)

newVal—The set value, specifying the code page for the host data file.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.XferHostCodePage

If (iVal=0) Then

'Set value

SessXfr.XferHostCodePage = 6

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_XferHostCodePage (&sVal);

if (sVal==0)

{

sVal = 6;

pXfr->put_XferHostCodePage (sVal);

}
```

Property: IHESessionFileTransfer::XferPCCodePage 3270

This property returns or sets a value specifying the code page for a data file on your computer. This value determines the location of non-ASCII characters such as é. Windows uses the ANSI (American National Standards Institute) code page; DOS systems use other code pages.

Basic Syntax

Integer = SessionFileTransfer.**XferPCCodePage**
SessionFileTransfer.**XferPCCodePage** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XferPCCodePage**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_XferPCCodePage**([in] short *newVal*);

Parameters

pVal—One of the following returned values, indicating the code page for a data file on your computer:

- 0—ANSI (Latin-1) (1252/819)
- 1—English US (437)
- 2—Canadian French (863)
- 3—Western European/Latin 1 (850)

newVal—The set value, specifying the code page for a data file on your computer.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.XferPCCodePage

If (iVal=0) Then

'Set value

SessXfr.XferPCCodePage = 2

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_XferPCCodePage (&sVal);

if (sVal==0)

{

sVal = 2;

pXfr->put_XferPCCodePage (sVal);

}
```

Property: IHESessionFileTransfer::XferSource 3270

This property returns or sets a value indicating whether a file has been transferred from a file system or the clipboard.

Basic Syntax

Integer = SessionFileTransfer.**XferSource**
SessionFileTransfer.**XferSource** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XferSource**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_XferSource**([in] short *newVal*);

Parameters

pVal—The following returned values specify where a file has been transferred from:

· 430—File system

· 431—Clipboard

newVal—The set value, specifying where a file has been transferred from.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.XferSource

If (iVal=430) Then

'Set value

SessXfr. XferSource = 431

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_XferSource (&sVal);

if (sVal==430)

{

sVal = 431;

pXfr->put_XferSource (sVal);

}
```

Property: IHESessionFileTransfer::XferStartAction 3270 5250 VT

This property returns or sets a value indicating how HostExplorer performs before uploading or downloading a file.

Basic Syntax

Integer = SessionFileTransfer.**XferStartAction**
SessionFileTransfer.**XferStartAction** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XferStartAction**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_XferStartAction**([in] short *newVal*);

Parameters

pVal—The following returned values indicate how HostExplorer performs before uploading or downloading a file:

· 200—No Action

· 201—Home

· 202—Enter

· 203—Clear

newVal—The set value, which indicates how HostExplorer performs before uploading or downloading a file.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.XferStartAction

If (iVal=200) Then

'Set value

SessXfr.XferStartAction = 203

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_XferStartAction (&sVal);

if (sVal==200)

{

sVal = 203;

pXfr->put_XferStartAction(sVal);

}
```

Property: IHESessionFileTransfer::Xm1KPacket VT

This property returns or sets a value indicating whether HostExplorer uses 1024 or 128 bytes as the packet size. Using 1024 bytes is good practice because it is faster. You may find the 128-byte setting useful if you have a communication problem such as a noisy telephone line.

Basic Syntax

Boolean = SessionFileTransfer.**Xm1KPacket**

SessionFileTransfer.**Xm1KPacket** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_Xm1KPacket**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_Xm1KPacket**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer uses 1024 bytes as the packet size. A returned value of VARIANT_FALSE indicates that HostExplorer uses 128 bytes as the packet size.

newVal—A value of VARIANT_TRUE indicates that HostExplorer uses 1024 bytes as the packet size. A value of VARIANT_FALSE indicates that HostExplorer uses 128 bytes as the packet size.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.Xm1Kpacket

If (bVal = False) Then

'Set value

SessXfr.Xm1Kpacket = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_Xm1Kpacket (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_Xm1Kpacket (bVal);

}
```

Property: IHESessionFileTransfer::XmAckTimeout VT

This property returns or sets a value specifying the length of time in milliseconds before a file transfer times out. Transfer delays are common. If a delay becomes too long, HostExplorer cancels the transfer, allowing you to try again later.

Basic Syntax

Long = SessionFileTransfer.**XmAckTimeout**
SessionFileTransfer.**XmAckTimeout** = Long

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XmAckTimeout**([out, retval] long * *pVal*);
HRESULT IHESessionFileTransfer::**put_XmAckTimeout**([in] long *newVal*);

Parameters

pVal—The returned value, specifying the length of time before a file transfer times out.
newVal—The set value, specifying the length of time before a file transfer times out.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim lVal As Long

'Get value

lVal = SessXfr.XmAckTimeout

If (lVal=0) Then

'Set value

SessXfr.XmAckTimeout = 15000

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

long lVal;

pXfr->get_XmAckTimeout (&lVal);

if (lVal==0)

{

lVal = 15000;

pXfr->put_XmAckTimeout (lVal);

}
```

Property: IHESessionFileTransfer::XmCRC VT

This property returns or sets a value indicating whether HostExplorer enables CRC (Cyclical Redundancy Check) to detect errors.

Basic Syntax

Boolean = SessionFileTransfer.**XmCRC**
SessionFileTransfer.**XmCRC** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_XmCRC**([out, retval] VARIANT_BOOL *
pVal);
HRESULT IHESessionFileTransfer::**put_XmCRC**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables CRC. A returned value of VARIANT_FALSE indicates that HostExplorer uses Checksum to check errors. (Use Checksum only when communicating with older Xmodem products.)
newVal—A value of VARIANT_TRUE indicates that HostExplorer enables CRC. A value of VARIANT_FALSE indicates that HostExplorer uses Checksum to check errors.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.XmCRC

If (bVal = False) Then

'Set value

SessXfr.XmCRC = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_XmCRC (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_XmCRC (bVal);

}
```

Property: IHESessionFileTransfer::YmAckTimeout VT

This property returns or sets a value specifying the length of time in milliseconds before a file transfer times out. Transfer delays are common. If a delay becomes too long, HostExplorer cancels the transfer, allowing you to try again later.

Basic Syntax

```
Long = SessionFileTransfer.YmAckTimeout
SessionFileTransfer.YmAckTimeout = Long
```

C++ Syntax

```
HRESULT IHESessionFileTransfer::get_YmAckTimeout([out, retval] long * pVal);
HRESULT IHESessionFileTransfer::put_YmAckTimeout([in] long newVal);
```

Parameters

pVal—The returned value, specifying the length of time before a file transfer times out.
newVal—The set value, specifying the length of time before a file transfer times out.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim lVal As Long

'Get value

lVal = SessXfr.YmAckTimeout

If (lVal=0) Then

'Set value

SessXfr.YmAckTimeout = 15000

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

long lVal;

pXfr->get_YmAckTimeout (&lVal);

if (lVal==0)

{

lVal = 15000;

pXfr->put_YmAckTimeout(lVal);

}
```

Property: IHESessionFileTransfer::YmUseFullPath VT

This property returns or sets a value indicating whether HostExplorer saves a file to the current UNIX host directory and lets you keep the entire path name as the file name.

Basic Syntax

Boolean = SessionFileTransfer.**YmUseFullPath**

SessionFileTransfer.**YmUseFullPath** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_YmUseFullPath**([out, retval] VARIANT_BOOL *
pVal);

HRESULT IHESessionFileTransfer::**put_YmUseFullPath**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer saves a file to the current UNIX host directory and keeps the entire pathname as the filename. A returned value of VARIANT_FALSE indicates that HostExplorer excludes the directory path from the file name.

newVal—A value of VARIANT_TRUE indicates that HostExplorer saves a file to the current UNIX host directory and keeps the entire pathname as the filename. A value of VARIANT_FALSE indicates that HostExplorer excludes the directory path from the file name.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.YmUseFullPath

If (bVal = False) Then

'Set value

SessXfr.YmUseFullPath = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_YmUseFullPath (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_YmUseFullPath (bVal);

}
```

Property: IHESessionFileTransfer::ZmAutoDownload VT

This property returns or sets a value indicating whether HostExplorer detects the initial header of a request to receive files and automatically starts the receiving process.

Basic Syntax

Boolean = SessionFileTransfer.**ZmAutoDownload**

SessionFileTransfer.**ZmAutoDownload** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ZmAutoDownload**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_ZmAutoDownload**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer detects the initial header of a request to receive files and automatically starts the receiving process. A returned value of VARIANT_FALSE indicates that HostExplorer does not detect the initial header of a request to receive files and does not automatically start the receiving process.

newVal—A value of VARIANT_TRUE indicates that HostExplorer detects the initial header of a request to receive files and automatically starts the receiving process. A value of VARIANT_FALSE indicates that HostExplorer does not detect the initial header of a request to receive files and does not automatically start the receiving process.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.ZmAutoDownload

If (bVal = False) Then

'Set value

SessXfr.ZmAutoDownload = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_ZmAutoDownload (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_ZmAutoDownload (bVal);

}
```

Property: IHESessionFileTransfer::ZmCrashRecovery VT

This property returns or sets a value indicating whether HostExplorer sets a computer to automatically resume receiving files if the computer crashed when the files were being transferred.

Basic Syntax

Boolean = SessionFileTransfer.**ZmCrashRecovery**

SessionFileTransfer.**ZmCrashRecovery** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ZmCrashRecovery**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_ZmCrashRecovery**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer sets a computer to automatically resume receiving files. A returned value of VARIANT_FALSE indicates that a computer does not automatically resume receiving files.

newVal—A value of VARIANT_TRUE indicates that HostExplorer sets a computer to automatically resume receiving files. A value of VARIANT_FALSE indicates that a computer does not automatically resume receiving files.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.ZmCrashRecovery

If (bVal = False) Then

'Set value

SessXfr.ZmCrashRecovery = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_ZmCrashRecovery (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_ZmCrashRecovery (bVal);

}
```

Property: IHESessionFileTransfer::ZmMaxErrors VT

This property returns or sets a value indicating how many times HostExplorer attempts to recover from errors before canceling a file transfer.

Basic Syntax

Integer = SessionFileTransfer.**ZmMaxErrors**

SessionFileTransfer.**ZmMaxErrors** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ZmMaxErrors**([out, retval] short * *pVal*);

HRESULT IHESessionFileTransfer::**put_ZmMaxErrors**([in] short *newVal*);

Parameters

pVal—The returned value, which indicates how many times HostExplorer attempts to recover from errors before canceling a file transfer.

newVal—The set value, which indicates how many times HostExplorer attempts to recover from errors before canceling a file transfer.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.ZmMaxErrors

If (iVal= 0) Then

'Set value

SessXfr.ZmMaxErrors = 10

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_ZmMaxErrors (&sVal);

if (sVal==0)

{

sVal = 10;

pXfr->put_ZmMaxErrors (sVal);

}
```

Property: IHESessionFileTransfer::ZmOverwriteMngmt VT

This property returns or sets a value specifying how files are written to the host. In order for this feature to function properly, the receiving host must also support this feature.

Basic Syntax

Integer = SessionFileTransfer.**ZmOverwriteMngmt**
SessionFileTransfer.**ZmOverwriteMngmt** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ZmOverwriteMngmt**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_ZmOverwriteMngmt**([in] short *newVal*);

Parameters

pVal—One of the following returned values, specifying how files are written to the host:

· 0—Newer or longer

· 1—Append

· 2—Always overwrite

· 3—File size or date differ

· 4—Never overwrite

· 5—Newer

newVal—The set value, specifying how files are written to the host.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.ZmOverwriteMngmt

If (iVal= 2) Then

'Set value

SessXfr.ZmOverwriteMngmt = 4

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_ZmOverwriteMngmt (&sVal);

if (sVal==2)

{

sVal = 4;

pXfr->put_ZmOverwriteMngmt (sVal);

}
```

Property: IHESessionFileTransfer::ZmSlideWindow VT

This property returns or sets a value indicating whether HostExplorer uses the Sliding Window option to send all files simultaneously. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionFileTransfer.**ZmSlideWindow**

SessionFileTransfer.**ZmSlideWindow** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ZmSlideWindow**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_ZmSlideWindow**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer sends all files simultaneously using the Sliding Window option. A returned value of VARIANT_FALSE indicates that HostExplorer sends files without using the Sliding Window option.

newVal—A value of VARIANT_TRUE indicates that HostExplorer sends all files simultaneously using the Sliding Window option. A value of VARIANT_FALSE indicates that HostExplorer sends files without using the Sliding Window option.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.ZmSlideWindow

If (bVal = False) Then

'Set value

SessXfr.ZmSlideWindow = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_ZmSlideWindow (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_ZmSlideWindow (bVal);

}
```

Property: IHESessionFileTransfer::ZmUseFullPath VT

This property returns or sets a value indicating whether HostExplorer saves a file to the current UNIX host directory and lets you keep the entire pathname as the file name.

Basic Syntax

Boolean = SessionFileTransfer.**ZmUseFullPath**

SessionFileTransfer.**ZmUseFullPath** = Boolean

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ZmUseFullPath**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFileTransfer::**put_ZmUseFullPath**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A returned value of VARIANT_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory.

newVal—A value of VARIANT_TRUE indicates that HostExplorer saves a file to the current UNIX host directory, and keeps the entire pathname as the file name. A value of VARIANT_FALSE indicates that HostExplorer does not save a file to the current UNIX host directory.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim bVal As Boolean

'Get value

bVal = SessXfr.ZmUseFullPath

If (bVal = False) Then

'Set value

SessXfr.ZmUseFullPath = True

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

VARIANT_BOOL bVal;

pXfr->get_ZmUseFullPath (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pXfr->put_ZmUseFullPath (bVal);

}
```

Property: IHESessionFileTransfer::ZmWindowSize VT

This property returns or sets a value indicating the byte size of files transferred using the Sliding Window option. By default, HostExplorer sends 8192 bytes at a time, thereby allowing the receiver enough time to receive the data.

Basic Syntax

Integer = SessionFileTransfer.**ZmWindowSize**
SessionFileTransfer.**ZmWindowSize** = Integer

C++ Syntax

HRESULT IHESessionFileTransfer::**get_ZmWindowSize**([out, retval] short * *pVal*);
HRESULT IHESessionFileTransfer::**put_ZmWindowSize**([in] short *newVal*);

Parameters

pVal—The returned value, which indicates the byte size of files transferred using the Sliding Window option.
newVal—The set value, which indicates the byte size of files transferred using the Sliding Window option.

Basic Example

```
Dim SessXfr As HESessionFileTransfer

Set SessXfr = Session.FileTransfer

Dim iVal As Integer

'Get value

iVal = SessXfr.ZmWindowSize

If (iVal= 0) Then

'Set value

SessXfr.ZmWindowSize = 8192

End If
```

C++ Example

```
IHESessionFileTransfer *pXfr;

pSess->get_FileTransfer(&pXfr);

short sVal;

pXfr->get_ZmWindowSize(&sVal);

if (sVal==0)

{

sVal = 8192;

pXfr->put_ZmWindowSize(sVal);

}
```

SessionSecurity Interface

The SessionSecurity interface lets you set configuration settings related to security.

Properties

The SessionSecurity interface consists of the following properties:

Kerberos

KerberosAltName

KerberosEncryption

KerberosForwardTkt

KerberosVersion

SecurityOption

Property: IHESessionSecurity::Kerberos 3270 VT

This property returns or sets a value indicating whether HostExplorer uses Kerberos as a network authentication protocol.

Basic Syntax

Boolean = SessionSecurity.**Kerberos**

SessionSecurity.**Kerberos** = Boolean

C++ Syntax

HRESULT IHESessionSecurity::**get_Kerberos**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionSecurity::**put_Kerberos**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer uses Kerberos as a network authentication protocol. A returned value of VARIANT_FALSE indicates that HostExplorer does not use Kerberos as a network authentication protocol.

newVal—A value of VARIANT_TRUE indicates that HostExplorer uses Kerberos as a network authentication protocol. A value of VARIANT_FALSE indicates that HostExplorer does not use Kerberos as a network authentication protocol.

Basic Example

```
Dim SessSec As HESessionSecurity

Set SessSec = Session.Security

Dim bVal As Boolean

'Get value

bVal = SessSec.Kerberos

If (bVal = False) Then

'Set value

SessSec.Kerberos = True

End If
```

C++ Example

```
IHESessionSecurity *pSec;

pSess->get_Security(&pSec);

VARIANT_BOOL bVal;

pSec->get_Kerberos (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSec->put_Kerberos (bVal);

}
```

Property: IHESessionSecurity::KerberosAltName 3270 VT

This property returns or sets a string that specifies the alternate user name when you are setting Kerberos authentication.

Basic Syntax

String = SessionSecurity.**KerberosAltName**
SessionSecurity.**KerberosAltName** = String

C++ Syntax

HRESULT IHESessionSecurity::**get_KerberosAltName**([out, retval] BSTR * *pVal*);
HRESULT IHESessionSecurity::**put_KerberosAltName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the alternate Kerberos user name.
newVal—The set string, specifying the alternate Kerberos user name.

Basic Example

```
Dim SessSec As HESessionSecurity

Set SessSec = Session.Security

Dim str As String

'Get value

str = SessSec.KerberosAltName

If (Len(str) = 0) Then

'Set value

SessSec.KerberosAltName = "Camelot56"

End If
```

C++ Example

```
IHESessionSecurity *pSec;

pSess->get_Security(&pSec);

BSTR bstr;

pSec->get_KerberosAltName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr=SysAllocString(OLESTR("Camelot56"));

pSec->put_KerberosAltName (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionSecurity::KerberosEncryption 3270 VT

This property returns or sets a value that specifies whether you need to use Kerberos encryption for your current session.

Basic Syntax

Boolean = SessionSecurity.**KerberosEncryption**

SessionSecurity.**KerberosEncryption** = Boolean

C++ Syntax

HRESULT IHESessionSecurity::**get_KerberosEncryption**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionSecurity::**put_KerberosEncryption**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that you need to use Kerberos encryption for your current session. A returned value of VARIANT_FALSE indicates that you do not need to use Kerberos encryption for your current session.

newVal—A value of VARIANT_TRUE indicates that you need to use Kerberos encryption for your current session. A value of VARIANT_FALSE indicates that you do not need to use Kerberos encryption for your current session.

Basic Example

```
Dim SessSec As HESessionSecurity

Set SessSec = Session.Security

Dim bVal As Boolean

'Get value

bVal = SessSec.KerberosEncryption

If (bVal = False) Then

'Set value

SessSec.KerberosEncryption = True

End If
```

C++ Example

```
IHESessionSecurity *pSec;

pSess->get_Security(&pSec);

VARIANT_BOOL bVal;

pSec->get_KerberosEncryption(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSec->put_KerberosEncryption(bVal);

}
```

Property: IHESessionSecurity::KerberosForwardTkt 3270 VT

This property returns or sets a value that enables or disables Kerberos ticket forwarding.

Basic Syntax

Boolean = SessionSecurity.**KerberosForwardTkt**

SessionSecurity.**KerberosForwardTkt** = Boolean

C++ Syntax

HRESULT IHESessionSecurity::**get_KerberosForwardTkt**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionSecurity::**put_KerberosForwardTkt**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that Kerberos ticket forwarding is enabled. A returned value of VARIANT_FALSE indicates that Kerberos ticket forwarding is disabled.

newVal—A value of VARIANT_TRUE indicates that Kerberos ticket forwarding is enabled. A value of VARIANT_FALSE indicates that Kerberos ticket forwarding is disabled.

Basic Example

```
Dim SessSec As HESessionSecurity

Set SessSec = Session.Security

Dim bVal As Boolean

'Get value

bVal = SessSec.KerberosForwardTkt

If (bVal = False) Then

'Set value

SessSec.KerberosForwardTkt = True

End If
```

C++ Example

```
IHESessionSecurity *pSec;

pSess->get_Security(&pSec);

VARIANT_BOOL bVal;

pSec->get_KerberosForwardTkt (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSec->put_KerberosForwardTkt (bVal);

}
```

Property: IHESessionSecurity::KerberosVersion 3270 VT

This property returns or sets the Kerberos version for the current session.

Basic Syntax

Integer = SessionSecurity.**KerberosVersion**
SessionSecurity.**KerberosVersion** = Integer

C++ Syntax

HRESULT IHESessionSecurity::**get_KerberosVersion**([out, retval] short * *pVal*);
HRESULT IHESessionSecurity::**put_KerberosVersion**([in] short *newVal*);

Parameters

pVal—The returned value, which indicates the Kerberos version of the current session.
newVal—The set value, which indicates the Kerberos version of the current session.

Basic Example

```
Dim SessSec As HESessionSecurity

Set SessSec = Session.Security

Dim iVal As Integer

'Get value

iVal = SessSec.KerberosVersion

If (iVal=3) Then

'Set value

SessSec.KerberosVersion = 4

End If
```

C++ Example

```
IHESessionSecurity *pSec;

pSess->get_Security(&pSec);

short sVal;

pSec->get_KerberosVersion(&sVal);

if (sVal ==3)

{

sVal = 4;

pSec->put_KerberosVersion(sVal);

}
```

Property: IHESessionSecurity::SecurityOption

This property returns or sets a value that specifies the method for securing the traffic between the server and the client.

Basic Syntax

HOSTEX_SECURITY_OPTIONS = SessionSecurity.**SecurityOption**
SessionSecurity.**SecurityOption** = HOSTEX_SECURITY_OPTIONS

C++ Syntax

HRESULT IHESessionSecurity::**get_SecurityOption**([out, retval]
HOSTEX_SECURITY_OPTIONS * *pVal*);
HRESULT IHESessionSecurity::**put_SecurityOption**([in]
HOSTEX_SECURITY_OPTIONS *newVal*);

Parameters

pVal—The returned value, specifying the security option.
newVal—The set value, specifying the security method.

Basic Example

```
Dim Session As HESession

Dim Security As HESessionSecurity

Dim SecurityVal As HOSTEX_SECURITY_OPTIONS



Set Session =
CreateObject("HESession.HESession")

Set Security = Session.Security



SecurityVal = Security.SecurityOption
```

C++ Example

```
IHESession* pSession;

IHESessionSecurity* pSecurity;

HOSTEX_SECURITY_OPTIONS SecurityOption;



HRESULT hr = HECoCreateInstance(CLSID_HESession,

IID_IHESession,

(LPVOID*)&pSession,

"HESession");



if (SUCCEEDED(hr))

{

pSession->get_Security(&pSecurity);

pSecurity->get_SecurityOption(&SecurityOption);

}
```

SessionDisplay Interface

The SessionDisplay interface lets you set configuration settings related to screen display.

Properties

The SessionDisplay interface consists of the following properties:

AlwaysAutoskip

BlinkToItalic

ConvertNulls

DisplayRowCol

DisplayUpperCase

EnableWorkspaceBackgroundBitmap

InsertResetByAttn

MultiLineDelete

MultiLineInsert

NoLockKeyb

RespectNumeric

ShowHotspots

ShowNulls

StatusLineMode

VTDefColsPerScreen

VTDefLinesPerScreen

VTHostWritableStatusLine

VTISOColors

VTMaxScrollBufferSize

VTSaveAttribsInScrollback

VTSaveEraseScreens

VTScrollNoBlanks

WorkSpaceBackgroundBitmap

WorkspaceBackgroundColor

WorkspaceForegroundColor

Property: IHESessionDisplay::AlwaysAutoskip 3270 5250

This property returns or sets a value indicating whether HostExplorer defines the field attribute that follows an unprotected field on the screen. By default, this property is set to VARIANT_FALSE.

Basic Syntax	Boolean = SessionDisplay.**AlwaysAutoskip** SessionDisplay.**AlwaysAutoskip** = Boolean
C++ Syntax	HRESULT IHESessionDisplay::**get_AlwaysAutoskip**([out, retval] VARIANT_BOOL * *pVal*); HRESULT IHESessionDisplay::**put_AlwaysAutoskip**([in] VARIANT_BOOL *newVal*);
Parameters	*pVal*—A returned value of VARIANT_TRUE indicates that HostExplorer defines the field attribute that follows an unprotected field on the screen. A returned value of VARIANT_FALSE indicates that HostExplorer does not define the field attribute that follows an unprotected field on the screen.

newVal—A value of VARIANT_TRUE indicates that HostExplorer defines the field attribute that follows an unprotected field on the screen. A value of VARIANT_FALSE indicates that HostExplorer does not define the field attribute that follows an unprotected field on the screen.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.AlwaysAutoskip

If (bVal = False) Then

SessDisp.AlwaysAutoskip = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_AlwaysAutoskip(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_AlwaysAutoskip (bVal);

}
```

Property: IHESessionDisplay::BlinkToItalic VT

This property returns or sets a value indicating whether HostExplorer maps the Blink attribute to an italicized font. This property is independent of the cursor mode. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**BlinkToItalic**

SessionDisplay.**BlinkToItalic** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_BlinkToItalic**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_BlinkToItalic**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer maps the Blink attribute to an italicized font. A returned value of VARIANT_FALSE indicates that HostExplorer does not map the Blink attribute to an italicized font.

newVal—A value of VARIANT_TRUE indicates that HostExplorer maps the Blink attribute to an italicized font. A value of VARIANT_FALSE indicates that HostExplorer does not map the Blink attribute to an italicized font.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.BlinkToItalic

If (bVal = False) Then

SessDisp.BlinkToItalic = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_BlinkToItalic (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_BlinkToItalic (bVal);

}
```

Property: IHESessionDisplay::ConvertNulls 3270 5250

This property converts zeros (nulls) to blank characters in input fields when copying and pasting text. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**ConvertNulls**
SessionDisplay.**ConvertNulls** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_ConvertNulls**([out, retval] VARIANT_BOOL *
pVal);
HRESULT IHESessionDisplay::**put_ConvertNulls**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the computer converts nulls to blank characters in input fields. A returned value of VARIANT_FALSE indicates that the computer does not convert nulls to blank characters in input fields.

newVal—A value of VARIANT_TRUE indicates that the computer converts nulls to blank characters in input fields. A value of VARIANT_FALSE indicates that the computer does not convert nulls to blank characters in input fields.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.ConvertNulls

If (bVal = False) Then

SessDisp.ConvertNulls = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_ConvertNulls (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_ConvertNulls (bVal);

}
```

Property: IHESessionDisplay::DisplayRowCol VT

This property returns or sets a value indicating whether HostExplorer displays the row and column indicator in the right-hand corner of the OIA (Operator Information Area). By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionDisplay.**DisplayRowCol**

SessionDisplay.**DisplayRowCol** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_DisplayRowCol**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_DisplayRowCol**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays the row and column indicator in the OIA. A returned value of VARIANT_FALSE indicates that HostExplorer does not display the row and column indicator in the OIA.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays the row and column indicator in the OIA. A value of VARIANT_FALSE indicates that HostExplorer does not display the row and column indicator in the OIA.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.DisplayRowCol

If (bVal = False) Then

SessDisp.DisplayRowCol = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_DisplayRowCol (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_DisplayRowCol (bVal);

}
```

Property: IHESessionDisplay::DisplayUpperCase 3270 5250

This property returns or sets a value indicating whether HostExplorer displays all output in upper case. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**DisplayUpperCase**

SessionDisplay.**DisplayUpperCase** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_DisplayUpperCase**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_DisplayUpperCase**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays all output in upper case. A returned value of VARIANT_FALSE indicates that HostExplorer does not change the output.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays all output in upper case. A value of VARIANT_FALSE indicates that HostExplorer does not change the output.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.DisplayUpperCase

If (bVal = False) Then

SessDisp.DisplayUpperCase = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_DisplayUpperCase(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_DisplayUpperCase(bVal);

}
```

Property: IHESessionDisplay::EnableWorkspaceBackgroundBitmap 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer enables the bitmap for the background of the terminal screen.

Basic Syntax

Boolean = SessionDisplay.**EnableWorkspaceBackgroundBitmap**

SessionDisplay.**EnableWorkspaceBackgroundBitmap** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_EnableWorkspaceBackgroundBitmap**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_EnableWorkspaceBackgroundBitmap**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables the bitmap for the background of the terminal screen. A returned value of VARIANT_FALSE indicates that HostExplorer uses the default terminal screen.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables the bitmap for the background of the terminal screen. A value of VARIANT_FALSE indicates that HostExplorer uses the default terminal screen.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.EnableWorkspaceBackgroundBitmap

If (bVal = False) Then

SessDisp.EnableWorkspaceBackgroundBitmap = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp-> get_EnableWorkspaceBackgroundBitmap(&bVal;

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->

put_EnableWorkspaceBackgroundBitmap(bVal);

}
```

Property: IHESessionDisplay::InsertResetByAttn 3270

This property lets you determine when the Insert key is toggled on. By default, this property is set to VARIANT_TRUE.

Note: When this property is set to VARIANT_TRUE, the Insert key remains on until you press an action key.

Basic Syntax

Boolean = SessionDisplay.**InsertResetByAttn**

SessionDisplay.**InsertResetByAttn** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_InsertResetByAttn**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_InsertResetByAttn**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Insert key is toggled on. A returned value of VARIANT_FALSE indicates that the Insert key remains toggled on until you press the Reset key.

newVal—A value of VARIANT_TRUE indicates that the Insert key is toggled on. A value of VARIANT_FALSE indicates that the Insert key remains toggled on until you press the Reset key.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.InsertResetByAttn

If (bVal = False) Then

SessDisp.InsertResetByAttn = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_InsertResetByAttn(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_InsertResetByAttn (bVal);

}
```

Property: IHESessionDisplay::MultiLineDelete 3270 5250

This property returns or sets a value indicating whether the Delete and Backspace keys remove characters from the current and subsequent lines. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**MultiLineDelete**

SessionDisplay.**MultiLineDelete** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_MultiLineDelete**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_MultiLineDelete**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Delete and Backspace keys remove characters from the current and subsequent lines. A returned value of VARIANT_FALSE indicates that the Delete and Backspace keys remove characters only from the current line.

newVal—A value of VARIANT_TRUE indicates that the Delete and Backspace keys remove characters from the current and subsequent lines. A value of VARIANT_FALSE indicates that the Delete and Backspace keys remove characters only from the current line.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.MultiLineDelete

If (bVal = False) Then

SessDisp.MultiLineDelete = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_MultiLineDelete (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_MultiLineDelete (bVal);

}
```

Property: IHESessionDisplay::MultiLineInsert 3270 5250

This property returns or sets a value indicating whether the Insert key inserts characters on the current and subsequent lines. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionDisplay.**MultiLineInsert**

SessionDisplay.**MultiLineInsert** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_MultiLineInsert**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_MultiLineInsert**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Insert key inserts characters on the current and subsequent lines. A returned value of VARIANT_FALSE indicates that the Insert key inserts characters and stops when the cursor reaches the end of the current line.

newVal—A value of VARIANT_TRUE indicates that the Insert key inserts characters on the current and subsequent lines. A value of VARIANT_FALSE indicates that the Insert key inserts characters and stops when the cursor reaches the end of the current line.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.MultiLineInsert

If (bVal = False) Then

SessDisp.MultiLineInsert = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_MultiLineInsert(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_MultiLineInsert(bVal);

}
```

Property: IHESessionDisplay::NoLockKeyb 3270 5250

This property returns or sets a value indicating whether HostExplorer sends a Never Lock the Keyboard command to the host. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionDisplay.**NoLockKeyb**

SessionDisplay.**NoLockKeyb** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_NoLockKeyb**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_NoLockKeyb**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer sends a Never Lock The Keyboard command to the host. A returned value of VARIANT_FALSE indicates that HostExplorer does not send a Never Lock The Keyboard command to the host.

newVal—A value of VARIANT_TRUE indicates that HostExplorer sends a Never Lock The Keyboard command to the host. A value of VARIANT_FALSE indicates that HostExplorer does not send a Never Lock The Keyboard command to the host.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.NoLockKeyb

If (bVal = False) Then

SessDisp.NoLockKeyb = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_NoLockKeyb(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_NoLockKeyb(bVal);

}
```

Property: IHESessionDisplay::RespectNumeric 3270

This property returns or sets a value indicating whether HostExplorer forces data entry on numeric fields to be validated. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**RespectNumeric**

SessionDisplay.**RespectNumeric** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_RespectNumeric**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_RespectNumeric**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer forces data entry on numeric fields to be validated. A returned value of VARIANT_FALSE indicates that HostExplorer does not force data entry on numeric fields to be validated.

newVal—A value of VARIANT_TRUE indicates that HostExplorer forces data entry on numeric fields to be validated. A value of VARIANT_FALSE indicates that HostExplorer does not force data entry on numeric fields to be validated.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.RespectNumeric

If (bVal = False) Then

SessDisp.RespectNumeric = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_RespectNumeric(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_RespectNumeric(bVal);

}
```

Property: IHESessionDisplay::ShowHotspots 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer displays hotspots, or "hot text" on the terminal screen. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionDisplay.**ShowHotspots**

SessionDisplay.**ShowHotspots** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_ShowHotspots**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_ShowHotspots**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays hotspots on the terminal screen. A returned value of VARIANT_FALSE indicates that HostExplorer does not display hotspots on the terminal screen.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays hotspots on the terminal screen. A value of VARIANT_FALSE indicates that HostExplorer does not display hotspots on the terminal screen.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.ShowHotspots

If (bVal = False) Then

SessDisp.ShowHotspots = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_ShowHotspots (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_ShowHotspots (bVal);

}
```

Property: IHESessionDisplay::ShowNulls 3270 5250

This property returns or sets a value indicating whether HostExplorer displays null characters in unprotected fields as centered dots. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**ShowNulls**

SessionDisplay.**ShowNulls** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_ShowNulls**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_ShowNulls**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays null characters as centered dots. A returned value of VARIANT_FALSE indicates that HostExplorer displays null characters as spaces.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays null characters as centered dots. A value of VARIANT_FALSE indicates that HostExplorer displays null characters as spaces.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.ShowNulls

If (bVal = False) Then

SessDisp.ShowNulls = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_ShowNulls (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_ShowNulls (bVal);

}
```

Property: IHESessionDisplay::StatusLineMode 3270 5250 VT

This property returns or sets the type of status-line mode (either terminal or window).

Basic Syntax

HOSTEX_STATUS_LINE_MODE = SessionDisplay.**StatusLineMode**

SessionDisplay.**StatusLineMode** = HOSTEX_STATUS_LINE_MODE

C++ Syntax

HRESULT IHESessionDisplay::**get_StatusLineMode**([out, retval] HOSTEX_STATUS_LINE_MODE * *pVal*);

HRESULT IHESessionDisplay::**put_StatusLineMode**([in] HOSTEX_STATUS_LINE_MODE *newVal*);

Parameters

pVal—The returned value, which indicates the type of status-line mode.

newVal—The set value, which indicates the type of status-line mode.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim slMode As HOSTEX_STATUS_LINE_MODE

slMode = SessDisp.StatusLineMode

If (slMode =

HOSTEX_STATUS_LINE_MODE_NOSTATUSLINE)

Then

SessDisp.StatusLineMode =

HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

HOSTEX_STATUS_LINE_MODE slMode;

pDisp->get_StatusLineMode(&slMode);

if (slMode == HOSTEX_STATUS_LINE_MODE_NOSTATUSLINE)

{

slMode = HOSTEX_STATUS_LINE_MODE_TERMINALSTATUSLINE;

pDisp->put_StatusLineMode(slMode);

}
```

Related Topics

HOSTEX_STATUS_LINE_MODE Data Type

Property: IHESessionDisplay::VTDefColsPerScreen VT

This property returns or sets a value specifying the default screen width that HostExplorer uses when it launches a new session. The available screen widths are:

· 80 columns

· 132 columns

· Custom—between 80 and 200 columns

Basic Syntax

Integer = SessionDisplay.**VTDefColsPerScreen**
SessionDisplay.**VTDefColsPerScreen** = Integer

C++ Syntax

HRESULT IHESessionDisplay::**get_VTDefColsPerScreen**([out, retval] short * *pVal*);
HRESULT IHESessionDisplay::**put_VTDefColsPerScreen**([in] short *newVal*);

Parameters

pVal—The returned value, specifying the default screen width that HostExplorer uses when it launches a new session.
newVal—The set value, specifying the default screen width that HostExplorer uses when it launches a new session.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim iVal As Integer

iVal = SessDisp.VTDefColsPerScreen

If (iVal = 80) Then

SessDisp.VTDefColsPerScreen = 120

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

short sVal;

pDisp->

get_VTDefColsPerScreen(&sVal);

if (sVal == 80)

{

sVal = 120;

pDisp->

put_VTDefColsPerScreen (sVal);

}
```

Property: IHESessionDisplay::VTDefLinesPerScreen VT

This property returns or sets a value specifying the default screen height that HostExplorer uses when it launches a new session. By default, this property is set to 24.

Basic Syntax

```
Integer = SessionDisplay.VTDefLinesPerScreen
SessionDisplay.VTDefLinesPerScreen = Integer
```

C++ Syntax

```
HRESULT IHESessionDisplay::get_VTDefLinesPerScreen([out, retval] short * pVal);
HRESULT IHESessionDisplay::put_VTDefLinesPerScreen([in] short newVal);
```

Parameters

pVal—The returned value, specifying the default screen height.
newVal—The set value, specifying the default screen height.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim iVal As Integer

iVal = SessDisp.VTDefLinesPerScreen

If (iVal = 24) Then

SessDisp.VTDefLinesPerScreen = 40

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

short sVal;

pDisp->

get_VTDefLinesPerScreen (&sVal);

if (sVal == 24)

{

sVal = 40;

pDisp->

put_VTDefLinesPerScreen(sVal);

}
```

Property: IHESessionDisplay::VTHostWritableStatusLine VT

This property returns or sets a value indicating whether you want HostExplorer to display messages on the status line. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**VTHostWritableStatusLine**

SessionDisplay.**VTHostWritableStatusLine** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_VTHostWritableStatusLine**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_VTHostWritableStatusLine**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays messages on the status line. A returned value of VARIANT_FALSE indicates that HostExplorer does not display messages on the status line.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays messages on the status line. A value of VARIANT_FALSE indicates that HostExplorer does not display messages on the status line.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.VTHostWritableStatusLine

If (bVal = False) Then

SessDisp.VTHostWritableStatusLine = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->

get_VTHostWritableStatusLine(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->

put_VTHostWritableStatusLine(bVal);

}
```

Property: IHESessionDisplay::VTISOColors VT

This property returns or sets a value indicating whether HostExplorer enables support for ISO colors for ANSI (American National Standards Institute) color-escape sequences on VT100, VT101, VT220, VT320, and VT420 models. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**VTISOColors**

SessionDisplay.**VTISOColors** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_VTISOColors**([out, retval] VARIANT_BOOL *_pVal_);

HRESULT IHESessionDisplay::**put_VTISOColors**([in] VARIANT_BOOL _newVal_);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables support for ISO colors. A returned value of VARIANT_FALSE indicates that HostExplorer disables support for ISO colors.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables support for ISO colors. A value of VARIANT_FALSE indicates that HostExplorer disables support for ISO colors.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.VTISOColors

If (bVal = False) Then

SessDisp.VTISOColors = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_VTISOColors(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_VTISOColors(bVal);

}
```

Property: IHESessionDisplay::VTMaxScrollBufferSize VT

This property returns or sets the number of lines that HostExplorer maintains in the Scrollback buffer. This number ranges from 1 to 9,999. By default, this property is set to 100. To disable the Scrollback buffer, set the property to 0.

Basic Syntax

Integer = SessionDisplay.**VTMaxScrollBufferSize**
SessionDisplay.**VTMaxScrollBufferSize** = Integer

C++ Syntax

HRESULT IHESessionDisplay::**get_VTMaxScrollBufferSize**([out, retval] short * *pVal*);
HRESULT IHESessionDisplay::**put_VTMaxScrollBufferSize**([in] short *newVal*);

Parameters

pVal—The returned value, indicating the number of lines that HostExplorer maintains in the Scrollback buffer.
newVal—The set value, indicating the number of lines that HostExplorer maintains in the Scrollback buffer.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim iVal As Integer

iVal = SessDisp.VTMaxScrollBufferSize

If (iVal = 0) Then

SessDisp.VTMaxScrollBufferSize = 2000

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

short sVal;

pDisp->

get_VTMaxScrollBufferSize(&sVal);

if (sVal == 0)

{

sVal = 2000;

pDisp->

put_VTMaxScrollBufferSize(sVal);
```

```
}
```

Property: IHESessionDisplay::VTSaveAttribsInScrollback VT

This property returns or sets a value indicating whether HostExplorer saves the Telnet screen attributes in the Scrollback buffer. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**VTSaveAttribsInScrollback**

SessionDisplay.**VTSaveAttribsInScrollback** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_VTSaveAttribsInScrollback**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_VTSaveAttribsInScrollback**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer saves the Telnet screen attributes in the Scrollback buffer. A returned value of VARIANT_FALSE indicates that HostExplorer does not save the Telnet screen attributes in the Scrollback buffer.

newVal—A value of VARIANT_TRUE indicates that HostExplorer saves the Telnet screen attributes in the Scrollback buffer. A value of VARIANT_FALSE indicates that HostExplorer does not save the Telnet screen attributes in the Scrollback buffer.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.VTSaveAttribsInScrollback

If (bVal = False) Then

SessDisp.VTSaveAttribsInScrollback = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->

get_VTSaveAttribsInScrollback(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->

put_VTSaveAttribsInScrollback(bVal);

}
```

Property: IHESessionDisplay::VTSaveEraseScreens VT

This property returns or sets a value indicating whether HostExplorer saves the screen to the Scrollback buffer before performing the Erase-Screen Host command. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionDisplay.**VTSaveEraseScreens**

SessionDisplay.**VTSaveEraseScreens** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_VTSaveEraseScreens**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_VTSaveEraseScreens**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer saves the screen to the Scrollback buffer before performing the Erase-Screen Host command. A returned value of VARIANT_FALSE indicates that HostExplorer does not save the screen to the Scrollback buffer before performing the Erase-Screen Host command.

newVal—A value of VARIANT_TRUE indicates that HostExplorer saves the screen to the Scrollback buffer before performing the Erase-Screen Host command. A value of VARIANT_FALSE indicates that HostExplorer does not save the screen to the Scrollback buffer before performing the Erase-Screen Host command.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.VTSaveEraseScreens

If (bVal = False) Then

SessDisp.VTSaveEraseScreens = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_VTSaveEraseScreens (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_VTSaveEraseScreens (bVal);

}
```

Property: IHESessionDisplay::VTScrollNoBlanks VT

This property returns or sets a value indicating whether to prevent HostExplorer from adding blank lines to the Scrollback buffer. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionDisplay.**VTScrollNoBlanks**

SessionDisplay.**VTScrollNoBlanks** = Boolean

C++ Syntax

HRESULT IHESessionDisplay::**get_VTScrollNoBlanks**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionDisplay::**put_VTScrollNoBlanks**([in] BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer cannot add blank lines to the Scrollback buffer, so you can compress new lines to create more space in the buffer. A returned value of VARIANT_FALSE indicates that HostExplorer can add blank lines to the Scrollback buffer.

newVal—A value of VARIANT_TRUE indicates that HostExplorer cannot add blank lines to the Scrollback buffer, so you can compress new lines to create more space in the buffer. A value of VARIANT_FALSE indicates that HostExplorer can add blank lines to the Scrollback buffer.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.VTScrollNoBlanks

If (bVal = False) Then

SessDisp.VTScrollNoBlanks = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->get_VTScrollNoBlanks(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->put_VTScrollNoBlanks(bVal);

}
```

Property: IHESessionDisplay::WorkSpaceBackgroundBitmap 3270 5250 VT

This property returns or sets a string specifying the bitmap file for the background of the terminal screen.

Basic Syntax

String = SessionDisplay.**WorkSpaceBackgroundBitmap**
SessionDisplay.**WorkSpaceBackgroundBitmap** = String

C++ Syntax

HRESULT IHESessionDisplay::**get_WorkSpaceBackgroundBitmap**([out, retval] BSTR * *pVal*);
HRESULT IHESessionDisplay::**put_WorkSpaceBackgroundBitmap**([in] BSTR *newVal*);

Parameters

pVal—The returned value, specifying the bitmap file for the background of the terminal screen.
newVal—The set value, specifying the bitmap file for the background of the terminal screen.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim bVal As Boolean

bVal = SessDisp.WorkSpaceBackgroundBitmap

If (bVal = False) Then

SessDisp.WorkSpaceBackgroundBitmap = True

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

VARIANT_BOOL bVal;

pDisp->

get_WorkSpaceBackgroundBitmap(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pDisp->

put_WorkSpaceBackgroundBitmap(bVal);

}
```

Property: IHESessionDisplay::WorkspaceBackgroundColor 3270 5250 VT

This property returns or sets a value specifying the background color of the terminal screen in OLE_COLOR format.

Basic Syntax

OLE_COLOR = SessionDisplay.**WorkspaceBackgroundColor**
SessionDisplay.**WorkspaceBackgroundColor** = OLE_COLOR

C++ Syntax

HRESULT IHESessionDisplay::**get_WorkspaceBackgroundColor**([out, retval] OLE_COLOR * *pVal*);
HRESULT IHESessionDisplay::**put_WorkspaceBackgroundColor**([in] OLE_COLOR *newVal*);

Parameters

pVal—The returned value, specifying the background color of the terminal screen.
newVal—The set value, specifying the background color of the terminal screen.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim color As OLE_COLOR

color = SessDisp.WorkspaceBackgroundColor

‘White is hex FFFFFF which is 16777215 ‘when converted to Long

If (color = 16777215) Then

‘Change from white to Green, which

‘is hex FF00,which converts to

‘Long 65280

SessDisp.WorkspaceBackgroundColor =

65280

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

long lVal;

//Get value

pDisp

->get_WorkspaceBackgroundColor (&lVal);

//White is hex FFFFFF which is 16777215

//when converted to Long

if (lVal = 16777215)

{

//Change from white to Green, which

//is hex FF00,which converts to

//Long 65280

lVal = 65280;

pDisp

->put_WorkspaceBackgroundColor (lVal);

}
```

Related Topics

OLE_COLOR Data Type

Property: IHESessionDisplay::WorkspaceForegroundColor 3270 5250 VT

This property returns or sets a value specifying the foreground color of the terminal screen in OLE_COLOR format.

Basic Syntax

OLE_COLOR = SessionDisplay.**WorkspaceForegroundColor**

SessionDisplay.**WorkspaceForegroundColor** = OLE_COLOR

C++ Syntax

HRESULT IHESessionDisplay::**get_WorkspaceForegroundColor**([out, retval] OLE_COLOR * *pVal*);

HRESULT IHESessionDisplay::**put_WorkspaceForegroundColor**([in] OLE_COLOR *newVal*);

Parameters

pVal—The returned value, specifying the foreground color of the terminal screen.

newVal—The returned value, specifying the foreground color of the terminal screen.

Basic Example

```
Dim SessDisp As HESessionDisplay

Set SessDisp = Session.Display

Dim color As OLE_COLOR

color = SessDisp.WorkspaceForegroundColor

‘White is hex FFFFFF which is 16777215 ‘when converted to Long

If (color = 16777215) Then

‘Change from white to Green, which

‘is hex FF00,which converts to

‘Long 65280

SessDisp.WorkspaceForegroundColor =

65280

End If
```

C++ Example

```
IHESessionDisplay *pDisp;

pSess->get_Display(&pDisp);

long lVal;

'Get value

pDisp

->get_WorkspaceForegroundColor(&lVal);

//White is hex FFFFFF which is 16777215

//when converted to Long

if (lVal = 16777215)

{

//Change from white to Green, which

//is hex FF00,which converts to

//Long 65280

lVal = 65280;

pDisp

->put_WorkspaceForegroundColor(lVal);

}
```

Related Topics

OLE_COLOR Data Type

SessionCursor Interface

The SessionCursor interface lets you set configuration settings related to the cursor.

Properties

The SessionCursor interface consists of the following properties:

CursorMode

CursorType

VTMoveCursorOnMouseClick

Property: IHESessionCursor::CursorMode 3270 5250 VT

This property returns or sets a value specifying the cursor mode (solid or blink) to use for the current session.

Basic Syntax

Integer = SessionCursor.**CursorMode**
SessionCursor.**CursorMode** = Integer

C++ Syntax

HRESULT IHESessionCursor::**get_CursorMode**([out, retval] short * *pVal*);
HRESULT IHESessionCursor::**put_CursorMode**([in] short *newVal*);

Parameters

pVal—The following returned values specify the cursor mode:

· 0—Solid

· 1—Blink

newVal—The set value, specifying the cursor mode.

Basic Example

```
Dim SessCur As HESessionCursor

Set SessCur = Session.Cursor

Dim iVal As Integer

iVal = SessCur.CursorMode

If (iVal = 0) Then

SessCur.CursorMode = 1

End If
```

C++ Example

```
IHESessionCursor *pCursor;

pSess->get_Cursor(&pCursor);

short sVal;

pCursor->get_CursorMode(&sVal);

if (sVal == 0)

{

sVal = 1;

pCursor->put_CursorMode(sVal);

}
```

Property: IHESessionCursor::CursorType 3270 5250 VT

This property returns or sets a value specifying the cursor type (vertical bar, underscore, or block) to use for the current session.

Basic Syntax

Integer = SessionCursor.**CursorType**
SessionCursor.**CursorType** = Integer

C++ Syntax

HRESULT IHESessionCursor::**get_CursorType**([out, retval] short * *pVal*);
HRESULT IHESessionCursor::**put_CursorType**([in] short *newVal*);

Parameters

pVal—The following returned values specify the cursor type:

· 0—Vertical bar

· 1—Underscore

· 2—Block

newVal—The set value, specifying the cursor type.

Basic Example

```
Dim SessCur As HESessionCursor

Set SessCur = Session.Cursor

Dim iVal As Integer

iVal = SessCur.CursorType

If (iVal = 0) Then

SessCur.CursorType = 1

End If
```

C++ Example

```
IHESessionCursor *pCursor;

pSess->get_Cursor(&pCursor);

short sVal;

pCursor->get_CursorType(&sVal);

if (sVal == 0)

{

sVal = 1;

pCursor->put_CursorType (sVal);

}
```

Property: IHESessionCursor::VTMoveCursorOnMouseClick VT

This property lets you force HostExplorer to automatically move the cursor when you click the mouse. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionCursor.**VTMoveCursorOnMouseClick**
SessionCursor.**VTMoveCursorOnMouseClick** = Boolean

C++ Syntax

HRESULT IHESessionCursor::**get_VTMoveCursorOnMouseClick**([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHESessionCursor::**put_VTMoveCursorOnMouseClick**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically moves the cursor when you click the mouse. A returned value of VARIANT_FALSE indicates that HostExplorer does not automatically move the cursor when you click the mouse.

newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically moves the cursor when you click the mouse. A value of VARIANT_FALSE indicates that HostExplorer does not automatically move the cursor when you click the mouse.

Basic Example

```
Dim SessCur As HESessionCursor

Set SessCur = Session.Cursor

Dim iVal As Integer

Dim bVal As Boolean

bVal = SessCur.VTMoveCursorOnMouseClick

If (bVal = False) Then

SessCur.VTMoveCursorOnMouseClick = True

End If
```

C++ Example

```
IHESessionCursor *pCursor;

pSess->get_Cursor(&pCursor);

VARIANT_BOOL bVal;

pCursor->

get_VTMoveCursorOnMouseClick(&bVal);

if (bVal == VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pCursor->

put_VTMoveCursorOnMouseClick (bVal);

}
```

SessionFonts Interface

The SessionFonts interface lets you set configuration settings related to fonts.

Properties

The SessionFonts interface consists of the following properties:

DisplayBorder

ProportionalFonts

SaveFontOnExit

SnapFrameBack

SwitchScreenType

Property: IHESessionFonts::DisplayBorder 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer fills the empty space between the terminal window and the window frame with a gray border. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionFonts.**DisplayBorder**

SessionFonts.**DisplayBorder** = Boolean

C++ Syntax

HRESULT IHESessionFonts::**get_DisplayBorder**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFonts::**put_DisplayBorder**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer creates a gray border between the terminal window and the window frame. A returned value of VARIANT_FALSE indicates that HostExplorer does not display a border, so that the screen display closely matches any resized window.

newVal—A value of VARIANT_TRUE indicates that HostExplorer creates a gray border between the terminal window and the window frame. A value of VARIANT_FALSE indicates that HostExplorer does not display a border, so that the screen display closely matches any resized window.

Basic Example

```
Dim SessFonts As HESessionFonts

Set SessFonts = Session.Fonts

Dim bVal As Boolean

bVal = SessFonts.DisplayBorder

If (bVal = False) Then

SessFonts.DisplayBorder = True

End If
```

C++ Example

```
IHESessionFonts* pFonts;

pSess->get_Fonts(&pFonts);

VARIANT_BOOL bVal;

pFonts->get_DisplayBorder (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pFonts->put_DisplayBorder (bVal);

}
```

Property: IHESessionFonts::ProportionalFonts 3270 5250 VT

This property returns or sets a value indicating whether to force HostExplorer to keep all fonts within a normal aspect ratio. This property lets HostExplorer better match fonts to the current window size. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionFonts.**ProportionalFonts**

SessionFonts.**ProportionalFonts** = Boolean

C++ Syntax

HRESULT IHESessionFonts::**get_ProportionalFonts**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFonts::**put_ProportionalFonts**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE forces HostExplorer to keep all fonts within a normal aspect ratio. A returned value of VARIANT_FALSE allows HostExplorer to create extra-wide or extra-tall fonts.

newVal—A value of VARIANT_TRUE forces HostExplorer to keep all fonts within a normal aspect ratio. A value of VARIANT_FALSE allows HostExplorer to create extra-wide or extra-tall fonts.

Basic Example

```
Dim SessFonts As HESessionFonts

Set SessFonts = Session.Fonts

Dim bVal As Boolean

bVal = SessFonts.ProportionalFonts

If (bVal = False) Then

SessFonts.ProportionalFonts = True

End If
```

C++ Example

```
IHESessionFonts* pFonts;

pSess->get_Fonts(&pFonts);

VARIANT_BOOL bVal;

pFonts->get_ProportionalFonts (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pFonts->put_ProportionalFonts (bVal);

}
```

Property: IHESessionFonts::SaveFontOnExit 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer automatically saves any changes you make to either the font or the window size to the current session profile. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionFonts.**SaveFontOnExit**

SessionFonts.**SaveFontOnExit** = Boolean

C++ Syntax

HRESULT IHESessionFonts::**get_SaveFontOnExit**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFonts::**put_SaveFontOnExit**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer automatically saves any changes you make to either the font or the window size. A returned value of VARIANT_FALSE indicates that HostExplorer does not automatically save any changes you make to either the font or the window size.

newVal—A value of VARIANT_TRUE indicates that HostExplorer automatically saves any changes you make to either the font or the window size. A value of VARIANT_FALSE indicates that HostExplorer does not automatically save any changes you make to either the font or the window size.

Basic Example

```
Dim SessFonts As HESessionFonts

Set SessFonts = Session.Fonts

Dim bVal As Boolean

bVal = SessFonts.SaveFontOnExit

If (bVal = False) Then

SessFonts.SaveFontOnExit = True

End If
```

C++ Example

```
IHESessionFonts* pFonts;

pSess->get_Fonts(&pFonts);

VARIANT_BOOL bVal;

pFonts->get_SaveFontOnExit(&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pFonts->put_SaveFontOnExit(bVal);

}
```

Property: IHESessionFonts::SnapFrameBack 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer displays borders on the terminal screen.

Basic Syntax

Boolean = SessionFonts.**SnapFrameBack**

SessionFonts.**SnapFrameBack** = Boolean

C++ Syntax

HRESULT IHESessionFonts::**get_SnapFrameBack**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionFonts::**put_SnapFrameBack**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays borders on the terminal screen. A returned value of VARIANT_FALSE indicates that HostExplorer does not display borders on the terminal screen.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays borders on the terminal screen. A value of VARIANT_FALSE indicates that HostExplorer does not display borders on the terminal screen.

Basic Example

```
Dim SessFonts As HESessionFonts

Set SessFonts = Session.Fonts

Dim bVal As Boolean

bVal = SessFonts.SnapFrameBack

If (bVal = False) Then

SessFonts.SnapFrameBack = True

End If
```

C++ Example

```
IHESessionFonts* pFonts;

pSess->get_Fonts(&pFonts);

VARIANT_BOOL bVal;

pFonts->get_SnapFrameBack (&bVal);

if (bVal=VARIANT_FALSE)

{

bVal=VARIANT_TRUE;

pFonts->put_SnapFrameBack (bVal);

}
```

Property: IHESessionFonts::SwitchScreenType 3270 5250 VT

This property returns or sets a value specifying the type of information that HostExplorer retains when you resize the screen.

Basic Syntax

HOSTEX_SWITCHSCREENTYPE = SessionFonts.**SwitchScreenType**

SessionFonts.**SwitchScreenType** = HOSTEX_SWITCHSCREENTYPE

C++ Syntax

HRESULT IHESessionFonts::**get_SwitchScreenType**([out, retval] HOSTEX_SWITCHSCREENTYPE * *pVal*);

HRESULT IHESessionFonts::**put_SwitchScreenType**([in] HOSTEX_SWITCHSCREENTYPE *newVal*);

Parameters

pVal—The returned value, specifying the type of information that HostExplorer retains when you resize the screen.

newVal—The set value, specifying the type of information that HostExplorer retains when you resize the screen.

Basic Example

```
Dim SessFonts As HESessionFonts

Dim ScreenType As HOSTEX_SWITCHSCREENTYPE

Set SessFonts = Session.Fonts

ScreenType = SessFonts.SwitchScreenType

If(ScreenType=

HOSTEX_SWITCHSCREENTYPE_KEEPFONT) then

SessFonts.SwitchScreenType =

HOSTEX_SWITCHSCREENTYPE_KEEPOLDINFO

End If
```

C++ Example

```
IHESessionFonts* pFonts;
pSess->get_Fonts(&pFonts);
HOSTEX_SWITCHSCREENTYPE ScreenType;
pFonts->
get_SwitchScreenType (&ScreenType);
if (ScreenType =
HOSTEX_SWITCHSCREENTYPE_KEEPFONT)
{
ScreenType =
HOSTEX_SWITCHSCREENTYPE_KEEPOLDINFO;
pFonts->
put_SwitchScreenType (ScreenType);
}
```

Related Topics

HOSTEX_SWITCHSCREENTYPE Data Type

SessionPCPrint Interface

The SessionPCPrint interface lets you set configuration settings related to the PCPRINT program.

Properties

The SessionPCPrint interface consists of the following properties:

7171PrintMode

PrinterDeinit

PrinterInit

TprintMode

Property: IHESessionPCPrint::7171PrintMode 3270

This property returns or sets a value specifying what sequences the emulator searches for in PassThru mode.

Basic Syntax

Integer = SessionPCPrint.**7171PrintMode**
SessionPCPrint.**7171PrintMode** = Integer

C++ Syntax

HRESULT IHESessionPCPrint::**get_7171PrintMode**([out, retval] short * *pVal*);
HRESULT IHESessionPCPrint::**put_7171PrintMode**([in] short *newVal*);

Parameters

pVal—The following returned values specify the sequences that the emulator searches for in pass-through mode:

· 0—VT100

· 1—IBM 3164

newVal—The set value, specifying the sequences that the emulator searches for in pass-through mode.

Basic Example

```
Dim SessFonts As HESessionFonts

Set SessFonts = Session.Fonts

Dim iVal As Integer

iVal = SessFonts.7171PrintMode

If (iVal =0) Then

SessFonts.7171PrintMode = 1

End If
```

C++ Example

```
IHESessionPCPrint *pPCPrint;

pSess->get_PCPrint(&pPCPrint);

short sVal;

pPCPrint->get_7171PrintMode (&sVal);

if (sVal == 0)

{

sVal = 1;

pPCPrint->put_7171PrintMode (sVal);



}
```

Property: IHESessionPCPrint::PrinterDeinit 3270

This property returns or sets a string specifying the escape sequences that you can send to the printer at the end of a PCPRINT/TPRINT job. The string can contain up to 255 characters and is printer-specific.

Basic Syntax

String = SessionPCPrint.**PrinterDeinit**

SessionPCPrint.**PrinterDeinit** = String

C++ Syntax

HRESULT IHESessionPCPrint::**get_PrinterDeinit**([out, retval] BSTR * *pVal*);

HRESULT IHESessionPCPrint::**put_PrinterDeinit**([in] BSTR *newVal*);

Parameters

pVal—The returned de-initialization string that is sent to the printer for pass-through printing.

newVal—The set de-initialization string that is sent to the printer for pass-through printing.

Basic Example

```
Dim SessPCPrint As HESessionPCPrint

Set SessPCPrint = Session.PCPrint

Dim str As String

'Get value

str = SessPCPrint.PrinterDeinit

If (Len(str) = 0) Then

'Set value

SessPCPrint.PrinterDeinit=

“^M-AT&C1&D2Q0V1X4E1S0=0^M”

End If
```

C++ Example

```
IHESessionPCPrint *pPCPrint;

pSess->get_PCPrint(&pPCPrint);

BSTR bstr;

pPCPrint->get_PrinterDeinit(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR

("^M-AT&C1&D2Q0V1X4E1S0=0^M"));

pPCPrint->put_PrinterDeinit(bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPCPrint::PrinterInit 3270

This property returns or sets a string specifying the escape sequences that you can send to the printer at the beginning of a PCPRINT/TPRINT job. The string can contain up to 255 characters and is printer-specific.

Basic Syntax

String = SessionPCPrint.**PrinterInit**
SessionPCPrint.**PrinterInit** = String

C++ Syntax

HRESULT IHESessionPCPrint::**get_PrinterInit**([out, retval] BSTR * *pVal*);
HRESULT IHESessionPCPrint::**put_PrinterInit**([in] BSTR *newVal*);

Parameters

pVal—The returned initialization string that is sent to the printer for pass-through printing.
newVal—The set initialization string that is sent to the printer for pass-through printing.

Basic Example

```
Dim SessPCPrint As HESessionPCPrint

Set SessPCPrint = Session.PCPrint

Dim str As String

'Get value

str = SessPCPrint.PrinterInit

If (Len(str) = 0) Then

'Set value

SessPCPrint.PrinterInit =

“^M-NT&C1&89ROCYX84X4E1S0=7^M”

End If
```

C++ Example

```
IHESessionPCPrint *pPCPrint;

pSess->get_PCPrint(&pPCPrint);

BSTR bstr;

pPCPrint->get_PrinterInit (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR

("^M-NT&C1&89ROCYX84X4E1S0=7^M"));

pPCPrint->put_PrinterInit (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPCPrint::TprintMode 3270 5250 VT

This property returns or sets a value specifying the output destination for TPRINT.

Basic Syntax

```
Integer = SessionPCPrint.TprintMode
SessionPCPrint.TprintMode = Integer
```

C++ Syntax

```
HRESULT IHESessionPCPrint::get_TprintMode([out, retval] short * pVal);
HRESULT IHESessionPCPrint::put_TprintMode([in] short newVal);
```

Parameters

pVal—The following returned values specify the output destination for TPRINT:

· 425—The default Windows printer.

· 426—The printer connected to an LPT1 parallel port.

· 427—The printer connected to an LPT2 parallel port.

· 428—The printer connected to an LPT3 parallel port.

· 429—The Windows clipboard.

newVal—The set value, specifying the output destination for TPRINT.

Basic Example

```
Dim SessFonts As HESessionFonts

Set SessFonts = Session.Fonts

Dim iVal As Integer

iVal = SessFonts.TprintMode

If (iVal =426) Then

SessFonts.TprintMode = 425

End If
```

C++ Example

```
IHESessionPCPrint *pPCPrint;

pSess->get_PCPrint(&pPCPrint);

short sVal;

pPCPrint->get_TprintMode(&sVal);

if (sVal == 426)

{

sVal = 425;

pPCPrint->put_TprintMode(sVal);

}
```

SessionPrintScreen Interface

The SessionPrintScreen interface lets you set configuration settings for printing a screen.

Properties

The SessionPrintScreen interface consists of the following properties:

AddFormFeed

DisplayAbortDlg

DisplayPrintDlg

PrintBlackAndWhite

PrintBorder

PrinterDocname

PrinterFooter

PrinterHeader

PrintLocation

PrintOIA

PrintReversedColors

PrintScreenFontName

PrintScreenFontPointSize

UseSpecificPrinter

Property: IHESessionPrintScreen::AddFormFeed 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer disables automatic form feed after HostExplorer sends a "screen image" to the printer.

Basic Syntax

Boolean = SessionPrintScreen.**AddFormFeed**

SessionPrintScreen.**AddFormFeed** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_AddFormFeed**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_AddFormFeed**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer enables automatic form feed. A returned value of VARIANT_FALSE indicates that HostExplorer disables automatic form feed.

newVal—A value of VARIANT_TRUE indicates that HostExplorer enables automatic form feed. A value of VARIANT_FALSE indicates that HostExplorer disables automatic form feed.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.AddFormFeed

If (bVal = False) Then

PrintScrn.AddFormFeed = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_AddFormFeed(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->put_AddFormFeed (bVal);

}
```

Property: IHESessionPrintScreen::DisplayAbortDlg 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer displays the Abort dialog box while printing. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionPrintScreen.**DisplayAbortDlg**

SessionPrintScreen.**DisplayAbortDlg** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_DisplayAbortDlg**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_DisplayAbortDlg**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays the Abort dialog box. A returned value of VARIANT_FALSE indicates that HostExplorer does not display the Abort dialog box.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays the Abort dialog box. A value of VARIANT_FALSE indicates that HostExplorer does not display the Abort dialog box.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.DisplayAbortDlg

If (bVal = False) Then

PrintScrn.DisplayAbortDlg = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_DisplayAbortDlg (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->put_DisplayAbortDlg (bVal);

}
```

Property: IHESessionPrintScreen::DisplayPrintDlg 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer displays the Print Screen dialog box when it begins printing. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionPrintScreen.**DisplayPrintDlg**

SessionPrintScreen.**DisplayPrintDlg** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_DisplayPrintDlg**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_DisplayPrintDlg**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer displays the PrintScreen dialog box when it begins printing. A returned value of VARIANT_FALSE indicates that HostExplorer does not display the PrintScreen dialog box.

newVal—A value of VARIANT_TRUE indicates that HostExplorer displays the PrintScreen dialog box when it begins printing. A value of VARIANT_FALSE indicates that HostExplorer does not display the PrintScreen dialog box.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.DisplayPrintDlg

If (bVal = False) Then

PrintScrn.DisplayPrintDlg = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_DisplayPrintDlg(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->put_DisplayPrintDlg(bVal);

}
```

Property: IHESessionPrintScreen::PrintBlackAndWhite 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer forces black-and-white printing on color printers by automatically converting colors to gray-scale. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionPrintScreen.**PrintBlackAndWhite**

SessionPrintScreen.**PrintBlackAndWhite** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrintBlackAndWhite**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_PrintBlackAndWhite**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE forces black-and-white printing. A returned value of VARIANT_FALSE indicates color printing.

newVal—A value of VARIANT_TRUE forces black-and-white printing. A value of VARIANT_FALSE indicates color printing.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.PrintBlackAndWhite

If (bVal = False) Then

PrintScrn.PrintBlackAndWhite = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_PrintBlackAndWhite(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->

put_PrintBlackAndWhite(bVal);

}
```

Property: IHESessionPrintScreen::PrintBorder 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prints a thick border around the printed screen image. You can set this property only if you specify Centered On Page in the PrintLocation property.

Basic Syntax

Boolean = SessionPrintScreen.**PrintBorder**

SessionPrintScreen.**PrintBorder** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrintBorder**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_PrintBorder**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer prints a thick border around the printed screen image. A returned value of VARIANT_FALSE indicates that HostExplorer does not print a border.

newVal—A value of VARIANT_TRUE indicates that HostExplorer prints a thick border around the printed screen image. A value of VARIANT_FALSE indicates that HostExplorer does not print a border.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.PrintBorder

If (bVal = False) Then

PrintScrn.PrintBorder = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_PrintBorder(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->put_PrintBorder(bVal);

}
```

Property: IHESessionPrintScreen::PrinterDocname 3270 5250 VT

This property returns or sets a string specifying the name of the document to be printed.

Basic Syntax

String = SessionPrintScreen.**PrinterDocname**
SessionPrintScreen.**PrinterDocname** = String

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrinterDocname**([out, retval] BSTR * *pVal*);
HRESULT IHESessionPrintScreen::**put_PrinterDocname**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the name of the document to be printed.
newVal—The set string, specifying the name of the document to be printed.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim str As String

str = PrintScrn.PrinterDocname

If (Len(str) = 0) Then

PrintScrn.PrinterDocname = “MyDoc.txt”

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

BSTR bstr;

pPrtScrn->get_PrinterDocname(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("MyDoc.txt"));

pPrtScrn->put_PrinterDocname(bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPrintScreen::PrinterFooter 3270 5250 VT

This property returns or sets a string specifying the information to appear in the document footer.

Basic Syntax

String = SessionPrintScreen.**PrinterFooter**
SessionPrintScreen.**PrinterFooter** = String

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrinterFooter**([out, retval] BSTR * *pVal*);
HRESULT IHESessionPrintScreen::**put_PrinterFooter**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the information to appear in the document footer.
newVal—The set string, specifying the information to appear in the document footer.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim str As String

str = PrintScrn.PrinterFooter

If (Len(str) = 0) Then

PrintScrn.PrinterFooter = “Hummingbird”

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

BSTR bstr;

pPrtScrn->get_PrinterFooter(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("Hummingbird"));

pPrtScrn->put_PrinterFooter(bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPrintScreen::PrinterHeader 3270 5250 VT

This property returns or sets a value specifying the information to appear in the document header.

Basic Syntax

String = SessionPrintScreen.**PrinterHeader**
SessionPrintScreen.**PrinterHeader** = String

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrinterHeader**([out, retval] BSTR * *pVal*);
HRESULT IHESessionPrintScreen::**put_PrinterHeader**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the information to appear in the document header.
newVal—The set string, specifying the information to appear in the document header.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim str As String

str = PrintScrn.PrinterHeader

If (Len(str) = 0) Then

PrintScrn.PrinterHeader = “Hummingbird”

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

BSTR bstr;

pPrtScrn->get_PrinterHeader (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("Hummingbird"));

pPrtScrn->put_PrinterHeader (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPrintScreen::PrintLocation 3270 5250 VT

This property returns or sets a value specifying how a screen appears on a printed page (either in the center or in the upper left-hand corner).

Basic Syntax

Integer = SessionPrintScreen.**PrintLocation**
SessionPrintScreen.**PrintLocation** = Integer

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrintLocation**([out, retval] short * *pVal*);
HRESULT IHESessionPrintScreen::**put_PrintLocation**([in] short *newVal*);

Parameters

pVal—The following returned values specify how a screen appears on a printed page:

· 485—Centered

· 486—Upper left

newVal—The set value, specifying how a screen appears on a printed page.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn=Session.PrintScreen

Dim iVal As Integer

iVal = PrintScrn.PrintLocation

If (iVal=486) Then

PrintScrn.PrintLocation = 485

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

short sVal;

pPrtScrn->

get_PrintLocation (&sVal);

if (sVal == 486)

{

sVal = 485;

pPrtScrn->

put_PrintLocation (sVal);

}
```

Property: IHESessionPrintScreen::PrintOIA 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prints the OIA (Operator Information Area) on the printed screen image.

Basic Syntax

Boolean = SessionPrintScreen.**PrintOIA**

SessionPrintScreen.**PrintOIA** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrintOIA**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_PrintOIA**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer prints the OIA on the printed screen image. A returned value of VARIANT_FALSE indicates that HostExplorer does not print the OIA on the printed screen image.

newVal—A value of VARIANT_TRUE indicates that HostExplorer prints the OIA on the printed screen image. A value of VARIANT_FALSE indicates that HostExplorer does not print the OIA on the printed screen image.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn=Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.PrintOIA

If (bVal = False) Then

PrintScrn.PrintOIA = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_PrintOIA(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->put_PrintOIA(bVal);

}
```

Property: IHESessionPrintScreen::PrintReversedColors 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prints reverse-color modes. This property forces the print engine to swap black and white colors while printing images. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionPrintScreen.**PrintReversedColors**

SessionPrintScreen.**PrintReversedColors** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrintReversedColors**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_PrintReversedColors**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer prints reverse-color modes. A returned value of VARIANT_FALSE indicates that HostExplorer does not print reverse-color modes.

newVal—A value of VARIANT_TRUE indicates that HostExplorer prints reverse-color modes. A value of VARIANT_FALSE indicates that HostExplorer does not print reverse-color modes.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn=Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.PrintReversedColors

If (bVal = False) Then

PrintScrn.PrintReversedColors = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_PrintReversedColors(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->

put_PrintReversedColors(bVal);

}
```

Property: IHESessionPrintScreen::PrintScreenFontName 3270 5250 VT

This property lets you change the default printer font used to print screen images (or the keyboard template) from Times New Roman to any TrueType font of your choice.

Basic Syntax

```
String = SessionPrintScreen.PrintScreenFontName
SessionPrintScreen.PrintScreenFontName = String
```

C++ Syntax

```
HRESULT IHESessionPrintScreen::get_PrintScreenFontName([out, retval] BSTR * pVal);
HRESULT IHESessionPrintScreen::put_PrintScreenFontName([in] BSTR newVal);
```

Parameters

pVal—The returned string, which indicates the TrueType font.
newVal—The set string, which indicates the TrueType font.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn=Session.PrintScreen

Dim str As String

str = PrintScrn.PrinterHeader

If (Len(str) = 0) Then

PrintScrn.PrintScreenFontName = “Arial”

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

BSTR bstr;

pPrtScrn->

get_PrintScreenFontName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =
```

```
SysAllocString(OLESTR("Arial"));

pPrtScrn->

put_PrintScreenFontName (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPrintScreen::PrintScreenFontPointSize 3270 5250 VT

This property returns or sets a value indicating the font size for printing.

Basic Syntax

Integer = SessionPrintScreen.**PrintScreenFontPointSize**
SessionPrintScreen.**PrintScreenFontPointSize** = Integer

C++ Syntax

HRESULT IHESessionPrintScreen::**get_PrintScreenFontPointSize**([out, retval] short * *pVal*);
HRESULT IHESessionPrintScreen::**put_PrintScreenFontPointSize**([in] short *newVal*);

Parameters

pVal—The returned value, which indicates the font size for printing.
newVal—The set value, which indicates the font size for printing.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim iVal As Integer

iVal = PrintScrn.PrintScreenFontPointSize

If (iVal=0) Then

PrintScrn.PrintScreenFontPointSize=12

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

short sVal;

pPrtScrn->

get_PrintScreenFontPointSize (&sVal);

if (sVal == 0)

{

sVal = 12;

pPrtScrn->

put_PrintScreenFontPointSize(sVal);

}
```

Property: IHESessionPrintScreen::UseSpecificPrinter 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer prints using the printer displayed in the Selected Printer Info box or the default printer. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionPrintScreen.**UseSpecificPrinter**

SessionPrintScreen.**UseSpecificPrinter** = Boolean

C++ Syntax

HRESULT IHESessionPrintScreen::**get_UseSpecificPrinter**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintScreen::**put_UseSpecificPrinter**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer prints using the specified printer. A returned value of VARIANT_FALSE indicates that HostExplorer prints using the default printer.

newVal—A value of VARIANT_TRUE indicates that HostExplorer prints using the specified printer. A value of VARIANT_FALSE indicates that HostExplorer prints using the default printer.

Basic Example

```
Dim PrintScrn As HESessionPrintScreen

Set PrintScrn = Session.PrintScreen

Dim bVal As Boolean

bVal = PrintScrn.UseSpecificPrinter

If (bVal = False) Then

PrintScrn.UseSpecificPrinter = True

End If
```

C++ Example

```
IHESessionPrintScreen *pPrtScrn;

pSess->get_PrintScreen(&pPrtScrn);

VARIANT_BOOL bVal;

pPrtScrn->get_UseSpecificPrinter(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrtScrn->

put_UseSpecificPrinter(bVal);

}
```

SessionPrintExplorer Interface

The SessionPrintExplorer interface lets you set configuration settings related to general output.

Properties

The SessionPrintExplorer interface consists of the following properties:

PEHostName

PELUName

PELUType

PEProfileName

PEStartPrinter

PEStopPrinter

Property: IHESessionPrintExplorer::PEHostName 3270 5250

This property returns or sets a string specifying the host name or IP address that PrintExplorer uses to establish a connection.

Basic Syntax

String = SessionPrintExplorer.**PEHostName**
SessionPrintExplorer.**PEHostName** = String

C++ Syntax

HRESULT IHESessionPrintExplorer::**get_PEHostName**([out, retval] BSTR * *pVal*);
HRESULT IHESessionPrintExplorer::**put_PEHostName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the host name or IP address.
newVal—The set string, specifying the host name or IP address.

Basic Example

```
Dim PrintEx As HESessionPrintExplorer

Set PrintEx = Session.PrintExplorer

Dim str As String

'Get value

str = PrintEx.PEHostName

If (Len(strVal) = 0) Then

'Set value

PrintEx.PEHostName = “vm1.mcgill.ca”

End If
```

C++ Example

```
IHESessionPrintExplorer *pPrintEx;

pSess->get_PrintExplorer(&pPrintEx);

BSTR bstr;

pPrintEx->

get_PEHostName(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("vm1.mcgill.ca"));

pPrintEx->

put_PEHostName(bstr);
```

```
SysFreeString(bstr);

}
```

Property: IHESessionPrintExplorer::PELUName 3270 5250

This property returns or sets a string specifying the LU (Logical Units) name for the current print session.

Basic Syntax

String = SessionPrintExplorer.**PELUName**
SessionPrintExplorer.**PELUName** = String

C++ Syntax

HRESULT IHESessionPrintExplorer::**get_PELUName**([out, retval] BSTR * *pVal*);
HRESULT IHESessionPrintExplorer::**put_PELUName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the LU name.
newVal—The set string, specifying the LU name.

Basic Example

```
Dim PrintEx As HESessionPrintExplorer

Set PrintEx = Session.PrintExplorer

Dim str As String

'Get value

str = PrintEx.PELUName

If (Len(strVal) = 0) Then

'Set value

PrintEx.PELUName = “LUNAme1”

End If
```

C++ Example

```
IHESessionPrintExplorer *pPrintEx;

pSess->get_PrintExplorer(&pPrintEx);

BSTR bstr;

pPrintEx->get_PELUName(&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("LUNAme1"));

pPrintEx->put_PELUName(bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPrintExplorer::PELUType 3270 5250

This property returns or sets a value specifying how the LU (Logical Units) type is determined.

Basic Syntax

Integer = SessionPrintExplorer.**PELUType**
SessionPrintExplorer.**PELUType** = Integer

C++ Syntax

HRESULT IHESessionPrintExplorer::**get_PELUType**([out, retval] short * *pVal*);
HRESULT IHESessionPrintExplorer::**put_PELUType**([in] short *newVal*);

Parameters

pVal—The following returned values indicate how the LU type is determined:

· 460—ConnectLU

· 461—LUName

· 462—AssociateLU

· 463—ProfileLU

newVal—The set string, which indicates how the LU type is determined.

Basic Example

```
Dim PrintEx As HESessionPrintExplorer

Set PrintEx = Session.PrintExplorer

Dim iVal As Integer

iVal = PrintEx.PELUType

If (iVal=462) Then

PrintEx.PELUType =464

End If
```

C++ Example

```
IHESessionPrintExplorer *pPrintEx;

pSess->get_PrintExplorer(&pPrintEx);

short sVal;

pPrintEx->get_PELUType (&sVal);

if (sVal == 462)

{

sVal = 464;

pPrintEx->put_PELUType (sVal);

}
```

Property: IHESessionPrintExplorer::PEProfileName 3270 5250

This property returns or sets a string specifying the LU (Logical Units) name for the Base PrintExplorer profile.

Basic Syntax

String = SessionPrintExplorer.**PEProfileName**
SessionPrintExplorer.**PEProfileName** = String

C++ Syntax

HRESULT IHESessionPrintExplorer::**get_PEProfileName**([out, retval] BSTR * *pVal*);
HRESULT IHESessionPrintExplorer::**put_PEProfileName**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the LU name for the Base PrintExplorer profile.
newVal—The set string, specifying the LU name for the Base PrintExplorer profile.

Basic Example

```
Dim PrintEx As HESessionPrintExplorer

Set PrintEx = Session.PrintExplorer

Dim str As String

'Get value

str = PrintEx.PEProfileName

If (Len(strVal) = 0) Then

'Set value

PrintEx.PEProfileName = “test.HPR”

End If
```

C++ Example

```
IHESessionPrintExplorer *pPrintEx;

pSess->get_PrintExplorer(&pPrintEx);

BSTR bstr;

pPrintEx->get_PEProfileName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("test.HPR "));

pPrintEx->put_PEProfileName(bstr);

SysFreeString(bstr);

}
```

Property: IHESessionPrintExplorer::PEStartPrinter 3270 5250

This property returns or sets a value indicating whether HostExplorer starts the printer automatically.

Basic Syntax

Boolean = SessionPrintExplorer.**PEStartPrinter**

SessionPrintExplorer.**PEStartPrinter** = Boolean

C++ Syntax

HRESULT IHESessionPrintExplorer::**get_PEStartPrinter**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintExplorer::**put_PEStartPrinter**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer starts the printer automatically. A returned value of VARIANT_FALSE indicates that HostExplorer does not start the printer automatically.

newVal—A value of VARIANT_TRUE indicates that HostExplorer starts the printer automatically. A value of VARIANT_FALSE indicates that HostExplorer does not start the printer automatically.

Basic Example

```
Dim PrintEx As HESessionPrintExplorer

Set PrintEx = Session.PrintExplorer

Dim bVal As Boolean

bVal = PrintEx. PEStartPrinter

If (bVal = False) Then

PrintEx. PEStartPrinter = True

End If
```

C++ Example

```
IHESessionPrintExplorer *pPrintEx;

pSess->get_PrintExplorer(&pPrintEx);

VARIANT_BOOL bVal;

pPrintEx->get_PEStartPrinter (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrintEx->put_PEStartPrinter (bVal);

}
```

Property: IHESessionPrintExplorer::PEStopPrinter 3270 5250

This property returns or sets a value indicating whether HostExplorer stops the printer.

Basic Syntax

Boolean = SessionPrintExplorer.**PEStopPrinter**

SessionPrintExplorer.**PEStopPrinter** = Boolean

C++ Syntax

HRESULT IHESessionPrintExplorer::**get_PEStopPrinter**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionPrintExplorer::**put_PEStopPrinter**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer stops the printer. A returned value of VARIANT_FALSE indicates that HostExplorer does not stop the printer.

newVal—A value of VARIANT_TRUE indicates that HostExplorer stops the printer. A value of VARIANT_FALSE indicates that HostExplorer does not stop the printer.

Basic Example

```
Dim PrintEx As HESessionPrintExplorer

Set PrintEx = Session.PrintExplorer

Dim bVal As Boolean

bVal = PrintEx.PEStopPrinter

If (bVal = False) Then

PrintEx.PEStopPrinter = True

End If
```

C++ Example

```
IHESessionPrintExplorer *pPrintEx;

pSess->get_PrintExplorer(&pPrintEx);

VARIANT_BOOL bVal;

pPrintEx->get_PEStopPrinter(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pPrintEx->put_PEStopPrinter(bVal);

}
```

SessionSaveFile Interface

The SessionSaveFile interface lets you set configuration settings related to the saving of files.

Properties

The SessionSaveFile interface consists of the following properties:

SaveAppend

SaveConfirm

SaveFileName

SaveMode

VTCaptureMode

Property: IHESessionSaveFile::SaveAppend 3270 5250 VT

This property returns or sets a value indicating whether HostExplorer adds a new file to the end of an existing file. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = SessionSaveFile.**SaveAppend**

SessionSaveFile.**SaveAppend** = Boolean

C++ Syntax

HRESULT IHESessionSaveFile::**get_SaveAppend**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionSaveFile::**put_SaveAppend**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer adds a new file to the end of an existing file. A returned value of VARIANT_FALSE indicates that HostExplorer does not add a new file to the end of an existing file.

newVal—A value of VARIANT_TRUE indicates that HostExplorer adds a new file to the end of an existing file. A returned value of VARIANT_FALSE indicates that HostExplorer does not add a new file to the end of an existing file.

Basic Example

```
Dim SessSave As HESessionSaveFile

Set SessSave = Session.SaveFile

Dim bVal As Boolean

bVal = SessSave.SaveAppend

If (bVal = False) Then

SessSave.SaveAppend = True

End If
```

C++ Example

```
IHESessionSaveFile *pSaveFile;

pSess->get_SaveFile(&pSaveFile);

VARIANT_BOOL bVal;

pSaveFile->get_SaveAppend(&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSaveFile->put_SaveAppend(bVal);

}
```

Property: IHESessionSaveFile::SaveConfirm 3270 5250 VT

This property returns or sets a value indicating whether to force the Save Screen command to prompt for a file name.

Basic Syntax

Boolean = SessionSaveFile.**SaveConfirm**

SessionSaveFile.**SaveConfirm** = Boolean

C++ Syntax

HRESULT IHESessionSaveFile::**get_SaveConfirm**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionSaveFile::**put_SaveConfirm**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE forces the Save Screen command to prompt for a file name. A returned value of VARIANT_FALSE does not force the Save Screen command to prompt for a file name.

newVal—A value of VARIANT_TRUE forces the Save Screen command to prompt for a file name. A value of VARIANT_FALSE does not force the Save Screen command to prompt for a file name.

Basic Example

```
Dim SessSave As HESessionSaveFile

Set SessSave = Session.SaveFile

Dim bVal As Boolean

bVal = SessSave.SaveConfirm

If (bVal = False) Then

SessSave.SaveConfirm = True

End If
```

C++ Example

```
IHESessionSaveFile *pSaveFile;

pSess->get_SaveFile(&pSaveFile);

VARIANT_BOOL bVal;

pSaveFile->get_SaveConfirm (&bVal);

if (bVal==VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pSaveFile->put_SaveConfirm (bVal);

}
```

Property: IHESessionSaveFile::SaveFileName 3270 5250 VT

This property returns or sets a string specifying the default path and file name that HostExplorer uses for saved and captured files. Unless otherwise specified, HostExplorer uses the same file name for all captured and saved screens.

Basic Syntax

```
String = SessionSaveFile.SaveFileName
SessionSaveFile.SaveFileName = String
```

C++ Syntax

```
HRESULT IHESessionSaveFile::get_SaveFileName([out, retval] BSTR * pVal);
HRESULT IHESessionSaveFile::put_SaveFileName([in] BSTR newVal);
```

Parameters

pVal—The returned string, specifying the default path and file name.
newVal—The set string, specifying the default path and file name.

Basic Example

```
Dim SessSave As HESessionSaveFile

Set SessSave = Session.SaveFile

Dim str As String

'Get value

str = SessSave.SaveFileName

If (Len(strVal) = 0) Then

'Set value

SessSave.SaveFileName = “test.SAV”

End If
```

C++ Example

```
IHESessionSaveFile *pSaveFile;

pSess->get_SaveFile(&pSaveFile);

BSTR bstr;

pSaveFile->get_SaveFileName (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("test.SAV"));

pSaveFile->put_SaveFileName (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionSaveFile::SaveMode 3270 5250 VT

This property returns or sets a value specifying the format in which HostExplorer saves the .SAV file.

Basic Syntax

```
Integer = SessionSaveFile.SaveMode
SessionSaveFile.SaveMode = Integer
```

C++ Syntax

```
HRESULT IHESessionSaveFile::get_SaveMode([out, retval] short * pVal);
HRESULT IHESessionSaveFile::put_SaveMode([in] short newVal);
```

Parameters

pVal—The following returned values specify the format for saved files:

· 447—ASCII

· 448—ANSI

newVal—The set value, specifying the save mode.

Basic Example

```
Dim SessSave As HESessionSaveFile

Set SessSave = Session.SaveFile

Dim iVal As Integer

'Get value

iVal = SessSave.Savemode

If (iVal= 448) Then

'Set value

SessSave.Savemode = 447

End If
```

C++ Example

```
IHESessionSaveFile *pSaveFile;

pSess->get_SaveFile(&pSaveFile);

short sVal;

pSaveFile->get_Savemode(&sVal);

if sVal==448)

{

sVal = 447;

pSaveFile->put_Savemode(sVal);

}
```

Property: IHESessionSaveFile::VTCaptureMode VT

This property returns or sets a value specifying how to capture selected text.

Basic Syntax

Integer = SessionSaveFile.**VTCaptureMode**
SessionSaveFile.**VTCaptureMode** = Integer

C++ Syntax

HRESULT IHESessionSaveFile::**get_VTCaptureMode**([out, retval] short * *pVal*);
HRESULT IHESessionSaveFile::**put_VTCaptureMode**([in] short *newVal*);

Parameters

pVal—The following returned values specify how to capture selected text:

· 546—Raw

· 547—Text

newVal—The set value, specifying how to capture selected text.

Basic Example

```
Dim SessSave As HESessionSaveFile

Set SessSave = Session.SaveFile

Dim iVal As Integer

'Get value

iVal = SessSave.VTCaptureMode

If (iVal= 546) Then

'Set value

SessSave.VTCaptureMode = 547

End If
```

C++ Example

```
IHESessionSaveFile *pSaveFile;

pSess->get_SaveFile(&pSaveFile);

short sVal;

pSaveFile->get_VTCaptureMode (&sVal);

if sVal==546)

{

sVal = 547;

pSaveFile->put_VTCaptureMode (sVal);

}
```

SessionVTPrint Interface

The SessionVTPrint interface lets you set configuration settings related to VTPrint.

Properties

The SessionVTPrint interface consists of the following properties:

VTPrinterTimeout

VTUseSpecificPrinter

Property: IHESessionVTPrint::VTPrinterTimeout VT

This property returns or sets a value specifying the delay (in seconds) before the printer prints a page.

Basic Syntax

```
Integer = SessionVTPrint.VTPrinterTimeout
SessionVTPrint.VTPrinterTimeout = Integer
```

C++ Syntax

```
HRESULT IHESessionVTPrint::get_VTPrinterTimeout([out, retval] short * pVal);
HRESULT IHESessionVTPrint::put_VTPrinterTimeout([in] short newVal);
```

Parameters

pVal—The returned value, specifying the printer delay.
newVal—The set value, specifying the printer delay.

Basic Example

```
Dim VTPrint As HESessionVTPrint

Set VTPrint= Session.VTPrint

Dim iVal As Integer

'Get value

iVal = VTPrint.VTPrinterTimeout

If (iVal= 0) Then

'Set value

VTPrint.VTPrinterTimeout = 500

End If
```

C++ Example

```
IHESessionVTPrint *pVTPrint;

pSess->get_VTPrint(&pVTPrint);

short sVal;

pVTPrint->get_VTPrinterTimeout(&sVal);

if sVal==0)

{

sVal = 500;

pVTPrint->put_VTPrinterTimeout(sVal);

}
```

Property: IHESessionVTPrint::VTUseSpecificPrinter VT

This property determines whether HostExplorer prints using the printer displayed in the Selected Printer Info box. When this option is disabled, HostExplorer uses the default printer. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = SessionVTPrint.**VTUseSpecificPrinter**

SessionVTPrint.**VTUseSpecificPrinter** = Boolean

C++ Syntax

HRESULT IHESessionVTPrint::**get_VTUseSpecificPrinter**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHESessionVTPrint::**put_VTUseSpecificPrinter**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that HostExplorer prints using the specified printer. A returned value of VARIANT_FALSE indicates that HostExplorer prints using the default printer.

newVal—A value of VARIANT_TRUE indicates that HostExplorer prints using the specified printer. A value of VARIANT_FALSE indicates that HostExplorer prints using the default printer.

Basic Example

```
Dim VTPrint As HESessionVTPrint

Set VTPrint= Session.VTPrint

Dim bVal As Boolean

'Get value

bVal = VTPrint.VTUseSpecificPrinter

If (bVal= False) Then

'Set value

VTPrint.VTUseSpecificPrinter = True

End If
```

C++ Example

```
IHESessionVTPrint *pVTPrint;

pSess->get_VTPrint(&pVTPrint);

VARIANT_BOOL bVal;

pVTPrint->

get_VTUseSpecificPrinter (&bVal);

if bVal == VARIANT_FALSE)

{

bVal = VARIANT_TRUE;

pVTPrint->

put_VTUseSpecificPrinter (bVal);

}
```

SessionSchemes Interface

The SessionSchemes interface lets you configure settings for a scheme (Color Schemes, File Transfer Schemes and Hotspot Schemes). A scheme is a collection of settings.

Properties

The SessionSchemes interface consists of the following properties:

Color

FileTransfer

Hotspot

Property: IHESessionSchemes::Color 3270 5250 VT

This property returns or sets a value indicating the color scheme to use for the current session.

Basic Syntax

String = SessionSchemes.**Color**
SessionSchemes.**Color** = String

C++ Syntax

HRESULT IHESessionSchemes::**get_Color**([out, retval] BSTR * *pVal*);
HRESULT IHESessionSchemes::**put_Color**([in] BSTR *newVal*);

Parameters

pVal—The returned value, which indicates the color scheme.
newVal—The set value, which indicates the color scheme.

Basic Example

```
Dim Schemes As HESessionSchemes

Set Schemes = Session.Schemes

Dim str As String

'Get value

str = Schemes.Color

If (Len(strVal) = 0) Then

'Set value

Schemes.Color = “color1.CS3”

End If
```

C++ Example

```
IHESessionSchemes *pSchemes;

pSess->get_Schemes(&pSchemes);

BSTR bstr;

pSchemes->get_Color (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("color1.CS3"));

pSchemes->put_Color (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionSchemes::FileTransfer 3270 5250 VT

This property returns or sets a string indicating the file-transfer scheme for the current session.

Basic Syntax

String = SessionSchemes.**FileTransfer**
SessionSchemes.**FileTransfer** = String

C++ Syntax

HRESULT IHESessionSchemes::**get_FileTransfer**([out, retval] BSTR * *pVal*);
HRESULT IHESessionSchemes::**put_FileTransfer**([in] BSTR *newVal*);

Parameters

pVal—The returned string, which indicates the file-transfer scheme.
newVal—The set string, which indicates the file-transfer scheme.

Basic Example

```
Dim Schemes As HESessionSchemes

Set Schemes = Session.Schemes

Dim str As String

'Get value

str = Schemes.FileTransfer

If (Len(strVal) = 0) Then

'Set value

Schemes.FileTransfer = “color1.CST”

End If
```

C++ Example

```
IHESessionSchemes *pSchemes;

pSess->get_Schemes(&pSchemes);

BSTR bstr;

pSchemes->get_FileTransfer (&bstr);

if (strlen(OLE2A(bstr))==0)

{

if (bstr!=NULL)

SysFreeString(bstr);

bstr =

SysAllocString(OLESTR("color1.CST"));

pSchemes->put_FileTransfer (bstr);

SysFreeString(bstr);

}
```

Property: IHESessionSchemes::Hotspot 3270 5250 VT

This property returns or sets a string indicating the hotspot scheme for the current session.

Basic Syntax

String = SessionSchemes.**Hotspot**
SessionSchemes.**Hotspot** = String

C++ Syntax

HRESULT IHESessionSchemes::**get_Hotspot**([out, retval] BSTR * *pVal*);
HRESULT IHESessionSchemes::**put_Hotspot**([in] BSTR *newVal*);

Parameters

pVal—The returned string, which indicates the hotspot scheme.
newVal—The set string, which indicates the hotspot scheme.

Basic Example

```
Dim Schemes As HESessionSchemes
Set Schemes = Session.Schemes
Dim str As String
'Get value
str = Schemes.Hotspot
If (Len(strVal) = 0) Then
'Set value
Schemes.Hotspot = “color1.CS3”
End If
```

C++ Example

```
IHESessionSchemes *pSchemes;
pSess->get_Schemes(&pSchemes);
BSTR bstr;
pSchemes->get_Hotspot(&bstr);
if (strlen(OLE2A(bstr))==0)
{
if (bstr!=NULL)
SysFreeString(bstr);
bstr =
SysAllocString(OLESTR("color1.CS3"));
pSchemes->put_Hotspot(bstr);
SysFreeString(bstr);
}
```

Data Types of the Session Object

The Session object contains the following data types:

HOSTEX_ATN_FORMAT

HOSTEX_CAPTURE_MODE

HOSTEX_CELL_DELIMITED

HOSTEX_CONNECT_BY

HOSTEX_CUT_MODE

HOSTEX_FIELD_ATTR_REPLACEMENT

HOSTEX_GRAPHICS_CELLSIZE

HOSTEX_GRAPHICS_CURSOR_TYPE

HOSTEX_GRAPHICS_MODEL

HOSTEX_INSERT_KEY_STYLE

HOSTEX_KEYBOARD_TYPE

HOSTEX_LINEMODE

HOSTEX_NEXT_FIELD_KEY

HOSTEX_PASTE_MODE

HOSTEX_SELECTION_MODE

HOSTEX_STATUS_LINE_MODE

HOSTEX_SWITCHSCREENTYPE

HOSTEX_TELNETECHO

HOSTEX_TERMINAL_ID

HOSTEX_TERM_MODEL

HOSTEX_ATN_FORMAT Data Type 3270 5250 VT

The HOSTEX_ATN_FORMAT data type consists of the type of sequences that you want to send to the host.

It has the following values:

HOSTEX_ATN_FORMAT_IBM	Indicates that the format is compatible with IBM emulators.
HOSTEX_ATN_FORMAT_WALLDATA	Indicates that the format is compatible with WallData emulators.
HOSTEX_ATN_FORMAT_ATTACHMATE	Indicates that the format is compatible with AttachMate® emulators.

Related Topics

Property: IHETransport::AttentionFormat

Property: IHESession::AttnFormat

About the Transport Objects

The Transport objects are tools for communication exchange and negotiation between your terminal and the host. The Transport objects receive and send data in EBCDIC format (for TN3270 and TN5250 terminals) or ASCII format (for TNVT terminals) from the host. This data is eventually displayed on the terminal.

The Transport objects consist of the following objects:

· HETP3270—A TN3270 terminal emulator that connects to an IBM mainframe.

· HETP5250—A TN5250 terminal emulator that connects to an AS/400 mainframe.

· HETPVT—A TNVT terminal emulator that connects to a UNIX or DEC machine.

· HETPSNA—An SNA terminal emulator that connects to a local server (Microsoft SNA), which connects you to the host.

· HETPSAA—An SAA terminal emulator that connects to a local server (Novell Netware), which connects you to the host.

· HETAPI—A telephone dial-up emulator that connects to a server.

For properties and/or data types specific to:

· only the HETP3270 object

· only the HETP5250 object

· only the HETPVT object

· both the HETP3270 and the VT objects

There are also methods and properties common to the HETP3270, HETP5250, and HETPVT objects

Related Topics

Methods of the Transport Objects

Properties of the Transport Objects

Data Types of the Transport Objects

Properties of the HETP3270 Object

The following properties are specific to the HETP3270 object:

EnableEMode

TNESession

Properties of the HETP5250 Object

The following properties are specific to the HETP5250 object:

DeviceName

Keyboard

MessageQueueLibrary

MessageQueueName

Password

Username

Properties and Data Types of the HETPVT Object

The following properties and data types are specific to the HETPVT object:

Properties

VTLineMode

Data Types

HOSTEX_LINEMODE

HOSTEX_TELNETECHO

Properties of the HETP3270/VT Objects

The following properties are specific to both the HETP3270 and the HETPVT objects:

EnableKerberosAuthentication

EnableKerberosEncryption

EnableKerberosTicketForwarding

IsEncrypted

KerberosAlternateUsername

KerberosUsername

KerberosVersion

Properties and Data Types of the HETP3270/5250/VT Objects

The following properties and data types are common to the HETP3270, HETP5250, and HETPVT objects:

Properties

AttentionFormat

CharSet

CodePage

Connected

ConnectionStatus

DeviceType

EnableFeatures

EnableTracing

HostName

IsReceiveBlocked

LockOnAttention

ModelColumns

ModelRows

Port

TelnetEcho

TelnetIsLineMode

TelnetisLocalEcho

TelnetName

TerminalModel

TraceFilename

Data Types

HOSTEX_ATN_FORMAT

HOSTEX_CON_STATUS

HOSTEX_DEVICE_TYPE

HOSTEX_FUNCTION_KEY

HOSTEX_TERM_MODEL

HOSTEX_TOGGLE_RECEIVE

Methods of the Transport Objects

The following properties are methods of the Transport objects:

AddFeature

GetStatusString

RemoveFeature

SendData

SendFunctionKey

SendKeepAlive

ToggleBlockReceive

Related Topics

Properties of the Transport Objects

Data Types of the Transport Objects

Method: IHETransport::AddFeature 3270 5250 VT

This method sets one property at a time in the Transport object.

Basic Syntax HETransport.**AddFeature**(*lFeature* As Long)

C++ Syntax HRESULT IHETransport::**AddFeature**(long *lFeature*);

Parameters *lFeature*—The feature that you set.

Basic Example

```
Dim Transport As HETransport

Dim lFeature As Long

Set Transport =
CreateObject("HETransport.HETransport")

lFeature = HOSTEX_TAB

Transport.AddFeature (lFeature)
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

m_pTransport->AddFeature(HOSTEX_EAB);
```

Method: IHETransport::GetStatusString 3270 5250 VT

This method specifies the status of the connection to the host.

Basic Syntax HETransport.**GetStatusString**(*lparam1* As Long, *1param2* As Long) As String

C++ Syntax HRESULT IHETransport::**GetStatusString**([in] long *lparam1*, [in] long *lParam2*, [out, retval] BSTR * *pStatus*);

Parameters *lparam1*—The user must set this parameter to 0.

1param2—The user must set this parameter to 0.

pStatus—The returned string, indicating the current connection status, or the last error that occurred.

Basic Example

```
Dim Transport As HETransport

Dim strStatus As String

Set Transport =
CreateObject("HETransport.HETransport")

strStatus = Transport.GetStatusString(0, 0)
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrStatus = NULL;

m_pTransport->GetStatusString

(0, 0, &bstrStatus);

SysFreeString(bstrStatus);
```

Method: IHETransport::RemoveFeature 3270 5250 VT

This method resets one property at a time in the Transport objects.

Basic Syntax HETransport.**RemoveFeature**(*lFeature* As Long)

C++ Syntax HRESULT IHETransport::**RemoveFeature**(long *lFeature*);

Parameters *lFeature*—The feature that is being removed or reset.

Basic Example

```
Dim Transport As HETransport

Dim lFeature As Long

Set Transport =
CreateObject("HETransport.HETransport")

lFeature = HOSTEX_TAB

Transport.RemoveFeature (lFeature)
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

m_pHETransport->RemoveFeature(HOSTEX_EAB);
```

Method: IHETransport::SendData 3270 5250 VT

This method sends the formatted data to the host.

Basic Syntax HETransport.**SendData**(*pBuffer* As String)

C++ Syntax HRESULT IHETransport::**SendData**(BSTR *pBuffer*);

Parameters *pBuffer*—The modified fields being sent.

Basic Example

```
Dim Transport As HETransport

Set Transport =
CreateObject("HETransport.HETransport")

Transport.SendData ("UserName")
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

m_pTransport->SendData("UserName");
```

Method: IHETransport::SendFunctionKey 3270 5250 VT

This method sends a particular function to the host.

Basic Syntax HETransport.**SendFunctionKey**(*IKey* As HOSTEX_FUNCTION_KEY)

C++ Syntax HRESULT IHETransport::**SendFunctionKey**(HOSTEX_FUNCTION_KEY *IKey*);

Parameters *IKey*—The function key that is sent to the host.

Basic Example

```
Dim Transport As HETransport

Set Transport =
CreateObject("HETransport.HETransport")

Transport.SendFunctionKey (HOSTEX_FUNCTION_KEY_SYSTEM_REQUEST)
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL, CLSCTX_INPROC_SERVER,
IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

m_pTransport->SendFunctionKey(HOSTEX_FUNCTION_KEY_SYSTEM_REQUEST)
```

Related Topics

HOSTEX_FUNCTION_KEY Data Type

Method: IHETransport::SendKeepAlive 3270 5250 VT

A repeated message that is sent to the host, indicating that you are still connected.

Basic Syntax HETransport.**SendKeepAlive**

C++ Syntax HRESULT IHETransport::**SendKeepAlive**();

Parameters This method has no parameters.

Basic Example

```
Dim Transport As HETransport

Set Transport =
CreateObject("HETransport.HETransport")

Transport.SendKeepAlive
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

m_pTransport->SendKeepAlive();
```

Method: IHETransport::ToggleBlockReceive 3270 5250 VT

This method toggles the blocking of the receipt of data from the Transport objects.

Basic Syntax HETransport.**ToggleBlockReceive**(*ToggleMode* As HOSTEX_TOGGLE_RECEIVE) As Boolean

C++ Syntax HRESULT IHETransport::**ToggleBlockReceive**([in] HOSTEX_TOGGLE_RECEIVE *ToggleMode*, [out, retval] VARIANT_BOOL * *pVal*);

Parameters *ToggleMode* —The toggle mode.

pVal—The returned value of the state after executing the method.

Basic Example

```
Dim Transport As HETransport

Dim bHoldState As Boolean

Set Transport =
CreateObject("HETransport.HETransport")

bHoldState = Transport.ToggleBlockReceive(HOSTEX_TOGGLE_RECEIVE_STATE)



If (bHoldState = True) Then

'Add code

Else

'Add code

End If
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL, CLSCTX_INPROC_SERVER,
IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bHoldState = VARIANT_FALSE;

m_transport->ToggleBlockReceive(HOSTEX_TOGGLE_RECEIVE_STATE, &bHoldState);



if(bHoldState == VARIANT_TRUE)

//Add code

else

//Add code
```

Properties of the Transport Objects

Properties define the characteristics of an object. The Transport objects have the following properties.

Note: These properties apply to all of the Transport objects unless specified in the property description.

AttentionFormat
CharSet
CodePage
Connected
ConnectionStatus
DeviceName
DeviceType
EnableEMode
EnableFeatures
EnableKerberosAuthentication
EnableKerberosEncryption
EnableKerberosTicketForwarding
EnableTracing
HostName
IsReceiveBlocked
KerberosAlternateUsername
KerberosUsername
KerberosVersion
Keyboard
LockOnAttention
MaxRecvBlockSize
MessageQueueLibrary
MessageQueueName
ModelColumns
ModelRows
Password
PerformingTransfer
Port
SecurityOption
SessionKeepAlive
TelnetEcho
TelnetIsLineMode
TelnetIsLocalEcho
TelnetName
TerminalModel
TNESession
TraceFilename
Username
VTLineMode

Related Topics

Methods of the Transport Objects

Data Types of the Transport Objects

Property: IHETransport::AttentionFormat 3270 5250 VT

This property returns or sets a value that enables compatibility with other servers or gateways. By default, this property is set to HOSTEX_ATN_FORMAT_ATTACHMATE.

Basic Syntax

HOSTEX_ATN_FORMAT = HETransport.**AttentionFormat**
HETransport.**AttentionFormat** = HOSTEX_ATN_FORMAT

C++ Syntax

HRESULT IHETransport::**get_AttentionFormat**([out, retval] HOSTEX_ATN_FORMAT * *pVal*);
HRESULT IHETransport::**put_AttentionFormat**([in] HOSTEX_ATN_FORMAT *newVal*);

Parameters

pVal—The returned value, which enables compatibility with other application formats.
newVal—The set value, which enables compatibility with other application formats.

Basic Example

```
Dim Transport As HETransport

Dim HostForamt As HOSTEX_ATN_FORMAT

Set Transport =
CreateObject("HETransport.HETransport")



'get attention format

HostFormat = Transport.AttentionFormat



'set attention format

HostFormat = HOSTEX_ATN_FORMAT_IBM

Transport.AttentionFormat = HostFormat
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

HOSTEX_ATN_FORMAT Format;

//get attention format

m_pTransport->get_AttentionFormat(&Format);



//put attention format

Format = HOSTEX_ATN_FORMAT_IBM;

m_pTransport->put_AttentionFormat(Format);
```

Related Topics

HOSTEX_ATN_FORMAT Data Type

Property: IHETransport::CharSet 5250

This property returns or sets character-set information to the host.

Basic Syntax

String = HETransport.**CharSet**
HETransport.**CharSet** = String

C++ Syntax

HRESULT IHETransport::**get_CharSet**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_CharSet**([in] BSTR *newVal*);

Parameters

pVal—The returned string, indicating the character-set information.
newVal—The set string, indicating the character-set information.

Basic Example

```
Dim Transport As HETransport

Dim strCharSet As String

Set Transport =
CreateObject("HETransport.HETransport")



'get code page

strCharSet = Transport.CharSet



'set code page

strCharSet = "CSID5250"

Transport.CharSet = strCharSet
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrCharSet = NULL;



//get char set

m_pTransport->get_CharSet(&bstrCharSet);



//put char set

bstrCharSet = SysAllocString(OLESTR("CSID5250”));

m_pTransport->put_AttentionFormat(bstrCharSet);



SysFreeString(bstrCharSet);
```

Related Topics

TN5250 Language-Conversion Table

Property: IHETransport::CodePage 5250

This property returns or sets a string that specifies what code page to use.

Basic Syntax

String = HETransport.**CodePage**
HETransport.**CodePage** = String

C++ Syntax

HRESULT IHETransport::**get_CodePage**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_CodePage**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the code page.
newVal—The set string, specifying the code page.

Basic Example

```
Dim Transport As HETransport

Dim strCodePage As String

Set Transport =
CreateObject("HETransport.HETransport")



'get code page

strCodePage = Transport.CodePage



'set code page

strCodePage = "English US"

Transport.CodePage = strCodePage
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrCodePage = NULL;



//get code page

m_pTransport->get_CodePage(&bstrCodePage);



//set code page

bstrCodePage = SysAllocString(OLESTR("English US”));
```

```
m_pTransport->put_AttentionFormat(bstrCodePage);


SysFreeString(bstrCodePage);
```

Related Topics

TN5250 Language-Conversion Table

Property: IHETransport::Connected 3270 5250 VT

This property returns or sets a value that indicates whether you are connected to the host.

Basic Syntax

```
Boolean = HETransport.Connected
HETransport.Connected = Boolean
```

C++ Syntax

```
HRESULT IHETransport::get_Connected([out, retval] VARIANT_BOOL * pVal);
HRESULT IHETransport::put_Connected([in] VARIANT_BOOL newVal);
```

Parameters

pVal—A returned value of VARIANT_TRUE indicates that you want to connect to the host. A returned value of VARIANT_FALSE indicates that you want to disconnect from the host.

newVal—A value of VARIANT_TRUE indicates that you want to connect to the host. A value of VARIANT_FALSE indicates that you want to disconnect from the host.

Basic Example

```
Dim Transport As HETransport

Dim bConnected As Boolean

Set Transport = CreateObject("HETransport.HETransport")



'get connected status

bConnected = Transport.Connected



'set connected status

bConnected = False

Transport.Connected = bConnected
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bConnected;
```

```
//get connected status

m_pTransport->get_Connected(&bConnected);



//put connected status

m_pTransport->put_Connected(VARIANT_FALSE);

m_pTransport->Release();

m_pTransport = NULL;
```

Property: IHETransport::ConnectionStatus 3270 5250 VT

This property returns a value that indicates the status of the connection to the host.

Basic Syntax
HOSTEX_CON_STATUS = HETransport.**ConnectionStatus**

C++ Syntax
HRESULT IHETransport::**get_ConnectionStatus**([out, retval] HOSTEX_CON_STATUS * *pVal*);

Parameters
pVal—The returned value, indicating the connection status to the host.

Basic Example

```
Dim Transport As HETransport

Dim ConnectStatus As HOSTEX_CON_STATUS

Set Transport = CreateObject("HETransport.HETransport")

ConnectStatus = Transport.ConnectionStatus
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

HOSTEX_CON_STATUS ConStat;

m_pHETransport->get_ConnectionStatus(&ConStat);
```

Related Topics

HOSTEX_CON_STATUS Data Type

Property: IHETransport::DeviceName 5250

This property returns or sets a value indicating whether you want to connect to a default or specific device name.

Basic Syntax

String = HETransport.**DeviceName**

HETransport.**DeviceName** = String

C++ Syntax

HRESULT IHETransport::**get_DeviceName**([out, retval] BSTR * *pVal*);

HRESULT IHETransport::**put_DeviceName**([in] BSTR *newVal*);

Parameters

pVal—The returned value, indicating whether you want to connect to a default or specific device name.

newVal—The set value, indicating whether you want to connect to a default or specific device name.

Basic Example

```
Dim Transport As HETransport

Dim strDeviceName As String

Set Transport = CreateObject("HETransport.HETransport")



'get deveice name

strDeviceName = Transport.DeviceName



'set device name

strDeviceName = "ValidName"

Transport.DeviceName = strDeviceName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrDeviceName = NULL;



//get device name

m_pTransport->get_DeviceName(&bstrDeviceName);



//put device name

bstrDeviceName = SysAllocString(OLESTR("ValidDeviceName”));

m_pTransport->put_DeviceName(bstrDeviceName);

SysFreeString(bstrDeviceName);
```

Property: IHETransport::DeviceType 3270 5250 VT

This property returns or sets a value that specifies whether the device type to be used with the Transport objects is a printer or display device.

Basic Syntax

HOSTEX_DEVICE_TYPE = HETransport.**DeviceType**

HETransport.**DeviceType** = HOSTEX_DEVICE_TYPE

C++ Syntax

HRESULT IHETransport::**get_DeviceType**([out, retval] HOSTEX_DEVICE_TYPE * *pVal*);

HRESULT IHETransport::**put_DeviceType**([in] HOSTEX_DEVICE_TYPE *newVal*);

Parameters

pVal—The returned value, indicating the device type to be used.

newVal—The set value, indicating the device type to be used.

Basic Example

```
Dim Transport As HETransport

Dim DeviceType As HOSTEX_DEVICE_TYPE

Set Transport = CreateObject("HETransport.HETransport")



'get device type

DeviceType = Transport.DeviceType



'set device type

DeviceType = HOSTEX_DEVICE_TYPE_DISPLAY

Transport.DeviceType = DeviceType
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

HOSTEX_DEVICE_TYPE DeviceType;



//get device type

m_pTransport->get_DeviceType(&DeviceType);



//put device type

deviceType = HOSTEX_DEVICE_TYPE_DISPLAY;

m_pTransport->put_DeviceType(DeviceType);
```

Related Topics

HOSTEX_DEVICE_TYPE Data Type

Property: IHETransport::EnableEMode 3270

This property returns or sets a value that enables the features for the TN3270E (Enhanced) terminal. By default, this property is set to VARIANT_TRUE.

Basic Syntax

Boolean = HETransport.**EnableEMode**
HETransport.**EnableEMode** = Boolean

C++ Syntax

HRESULT IHETransport::**get_EnableEMode** ([out, retval] VARIANT_BOOL * *pVal*);
HRESULT IHETransport::**put_EnableEMode** ([in] VARIANT_BOOL *newVal*);

Parameters

pVal—The returned value, which enables the features for the TN3270E terminal.
newVal—The set value, which enables the features for the TN3270E terminal.

Basic Example

```
Dim Transport As HETransport

Dim bMode As Boolean

Set Transport = CreateObject("HETransport.HETransport")



'get the mode

bMode = Transport.EnableEMode



'set the mode

bMode = False

Transport.EnableEMode = bMode
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL vbEMode;



//get mode

vbEmode = m_pTransport->get_EnableEMode();



//set mode

m_pTransport->put_EnableEMode(VARIANT_TRUE);
```

Property: IHETransport::EnableFeatures 3270 5250 VT

This property lets you set multiple transport properties simultaneously.

Basic Syntax

Long = HETransport.**EnableFeatures**
HETransport.**EnableFeatures** = Long

C++ Syntax

HRESULT IHETransport::**get_EnableFeatures**([out, retval] long * *pVal*);
HRESULT IHETransport::**put_EnableFeatures**([in] long *newVal*);

Parameters

pVal—The returned value, indicating the transport properties being set.
newVal—The set value, indicating the transport properties being set.

Basic Example

```
Dim Transport As HETransport

Dim lFeature As Long

Set Transport = CreateObject("HETransport.HETransport")



'get the feature

lFeature = Transport.EnableFeatures



'set the feature

lFeature = HOSTEX_SSCP_LU_MODE

Transport.EnableFeatures = lFeature
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

long lFeature;



//get enablefeatures

m_pTransport->get_EnableFeatures(&lFeature);



//put enablefeatures

lFeature = HOSTEX_SSCP_LU_MODE;

m_pTransport->put_ EnableFeatures (lFeature);
```

Property: IHETransport::EnableKerberosAuthentication 3270 VT

This property returns or sets a value that specifies whether you need to use Kerberos authentication for your current session. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HETransport. **EnableKerberosAuthentication**

HETransport.**EnableKerberosAuthentication** = Boolean

C++ Syntax

HRESULT IHETransport::**get_EnableKerberosAuthentication**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHETransport::**put_EnableKerberosAuthentication**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that you need to use Kerberos authentication. A returned value of VARIANT_FALSE indicates that you do not need to use Kerberos authentication.

newVal—A value of VARIANT_TRUE indicates that you need to use Kerberos authentication. A value of VARIANT_FALSE indicates that you do not need to use Kerberos authentication.

Basic Example

```
Dim Transport As HETransport

Dim bStatus As Boolean

Set Transport = CreateObject("HETransport.HETransport")



'get authentication status

bStatus = Transport.EnableKerberosAuthentication



'set authentication status

bStatus = False

Transport.EnableKerberosAuthentication = bStatus
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bVal;



//get kerberos authentication status

m_pTransport-> get_EnableKerberosAuthentication(&bVal);



//set kerberos authentication status
```

```
bVal = BOOL_FALSE;

m_pTransport-> put_EnableKerberosAuthentication(bVal);
```

Property: IHETransport::EnableKerberosEncryption 3270 VT

This property returns or sets a value that specifies whether you need to use Kerberos encryption for your current session. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HETransport.**EnableKerberosEncryption**

HETransport.**EnableKerberosEncryption** = Boolean

C++ Syntax

HRESULT IHETransport::**get_EnableKerberosEncryption**([out, retval] VARIANT_BOOL * pVal);

HRESULT IHETransport::**put_EnableKerberosEncryption**([in] VARIANT_BOOL newVal);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that you need to use Kerberos encryption for your current session. A returned value of VARIANT_FALSE indicates that you do not need to use Kerberos encryption for your current session.

newVal—A value of VARIANT_TRUE indicates that you need to use Kerberos encryption for your current session. A value of VARIANT_FALSE indicates that you do not need to use Kerberos encryption for your current session.

Basic Example

```
Dim Transport As HETransport

Dim bStatus As Boolean

Set Transport = CreateObject("HETransport.HETransport")



'get encryption status

bStatus = Transport.EnableKerberosEncryption



'set encryption status

bStatus = False

Transport.EnableKerberosEncryption = bStatus
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bVal;
```

```
//get kerberos encryption status

m_pTransport-> get_EnableKerberosEncryption(&bVal);



//set kerberos encryption status

bVal = BOOL_FALSE;

m_pTransport-> put_EnableKerberosEncryption(bVal);
```

Property: IHETransport::EnableKerberosTicketForwarding 3270 VT

This property returns or sets a value which enables or disables Kerberos ticket forwarding. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HETransport.**EnableKerberosTicketForwarding**

HETransport.**EnableKerberosTicketForwarding** = Boolean

C++ Syntax

HRESULT IHETransport::**get_EnableKerberosTicketForwarding**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHETransport::**put_EnableKerberosTicketForwarding**([in] VARIANT_BOOL * *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that Kerberos ticket forwarding is enabled. A returned value of VARIANT_FALSE indicates that Kerberos ticket forwarding is disabled.

newVal—A value of VARIANT_TRUE indicates that Kerberos ticket forwarding is enabled. A value of VARIANT_FALSE indicates that Kerberos ticket forwarding is disabled.

Basic Example

```
Dim Transport As HETransport

Dim bStatus As Boolean

Set Transport = CreateObject("HETransport.HETransport")



'get ticket forwarding status

bStatus = Transport.EnableKerberosTicketForwarding



'set ticket forwarding status

bStatus = False

Transport.EnableKerberosTicketForwarding = bStatus
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bVal;



//get kerberos ticket forwarding status

m_pTransport-> get_EnableKerberosTicketForwarding(&bVal);



//set kerberos ticket forwarding status

bVal = BOOL_FALSE;

m_pTransport-> put_EnableKerberosTicketForwarding(bVal);
```

Property: IHETransport::EnableTracing 3270 5250 VT

This property returns or sets a value that specifies whether you want to enable tracing. Tracing is the process of writing the information that is sent to and received from the host into a file. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HETransport.**EnableTracing**

HETransport.**EnableTracing** = Boolean

C++ Syntax

HRESULT IHETransport::**get_EnableTracing**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHETransport::**put_EnableTracing**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE enables tracing. A returned value of VARIANT_FALSE disables tracing.

newVal—A value of VARIANT_TRUE enables tracing. A value of VARIANT_FALSE disables tracing.

Basic Example

```
Dim Transport As HETransport

Dim bStatus As Boolean

Set Transport = CreateObject("HETransport.HETransport")



'get tracing status

bStatus = Transport.EnableTracing



'set tracing status

bStatus = False

Transport.EnableTracing = bStatus
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bTrace;



//get tracing status

m_pTransport->get_EnableTracing(&bTrace);



//set tracing status

bTrace = BOOL_FALSE;

m_pTransport->put_EnableTracing(bTrace);
```

Property: IHETransport::HostName 3270 5250 VT

This property returns or sets a string that specifies the name of the host to which you are trying to connect.

Basic Syntax

String = HETransport.**HostName**
HETransport.**HostName** = String

C++ Syntax

HRESULT IHETransport::**get_HostName**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_HostName**([in] BSTR *newVal*);

Parameters

pVal—The returned string indicating the host name.
newVal—The set string indicating the host name.

Basic Example

```
Dim Transport As HETransport

Dim strHostName As String

Set Transport = CreateObject("HETransport.HETransport")



'get host name

strHostName = Transport.HostName



'set host name

strHostName = "ValidName"

Transport.HostName = strHostName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrHost = NULL;



//get host name

m_pTransport->get_HostName(&bstrHost);



//set host name

bstrHost = SysAllocString(OLESTR("HostName”));

m_pTransport->put_HostName(bstrHost);



SysFreeString(bstrHost);
```

Property: IHETransport::IsEncrypted 3270 VT

This property returns the level of encryption of the message that is sent from the terminal to the host.

Basic Syntax HOSTEX_ENCRYPTED = HETransport.**IsEncrypted**

C++ Syntax HRESULT IHETransport::**get_IsEncrypted**([out, retval] HOSTEX_ENCRYPTED * *pVal*);

Parameters *pVal*—The returned value, which indicates the level of encryption.

Basic Example

```
Dim Transport As HETransport

Dim EncryptedStatus As HOSTEX_ENCRYPTED



Set Transport =
CreateObject("HETPVT.HETransportVT")



EncryptedStatus = Transport.IsEncrypted
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}



HOSTEX_ENCRYPTED EncryptedStatus;



m_pTransport->get_IsEncrypted(&EncryptedStatus);



switch(EncryptedStatus)

{

//add code

}
```

Property: IHETransport::IsReceiveBlocked 3270 5250 VT

This property returns the state of whether the receive is blocked.

Basic Syntax Boolean = HETransport.**IsReceiveBlocked**

C++ Syntax HRESULT IHETransport::**get_IsReceiveBlocked**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the receive is blocked. A returned value of VARIANT_FALSE indicates that the receive is not blocked.

Basic Example

```
Dim Transport As HETransport

Dim bStatus As Boolean

Set Transport = CreateObject("HETransport.HETransport")

bStatus = Transport.IsReceiveBlocked
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bReceived;

m_pTransport->IsReceiveBlocked(&bReceived);
```

Related Topics

HOSTEX_TOGGLE_RECEIVE Data Type

Property: IHETransport::KerberosAlternateUsername 3270 VT

This property returns or sets a string that specifies the alternate user name when Kerberos authentication is being set.

Basic Syntax String = HETransport.**KerberosAlternateUsername**
HETransport.**KerberosAlternateUsername** = String

C++ Syntax HRESULT IHETransport::**get_KerberosAlternateUsername**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_KerberosAlternateUsername**([in] BSTR * *newVal*);

Parameters *pVal*—The returned string, indicating the alternate Kerberos user name.
newVal—The set string, indicating the alternate Kerberos user name.

Basic Example

```
Dim Transport As HETransport

Dim strUserName As String

Set Transport = CreateObject("HETransport.HETransport")



'get alternate user name

strUserName = Transport.KerberosAlternateUsername



'set alternate user name

strUserName = "ValidName"

Transport.KerberosAlternateUsername = strUserName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL, CLSCTX_INPROC_SERVER,
IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrKerberosAltName = NULL;



//get kerberos alternate user name

m_pTransport->get_KerberosAlternateUsername(&bstrKerberosAltName);



//set kerberos alternate user name

bstrKerberosAltName = SysAllocString(OLESTR("ValidName”));

m_pTransport->put_KerberosAlternateUsername(bstrKerberosAltName);



SysFreeString(bstrKerberosAltName);
```

Property: IHETransport::KerberosUsername 3270 VT

This property returns or sets a value that specifies the user name when Kerberos authentication is being set.

Basic Syntax

String = HETransport.**KerberosUsername**
HETransport.**KerberosUsername** = String

C++ Syntax

HRESULT IHETransport::**get_KerberosUsername**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_KerberosUsername**([in] BSTR *newVal*);

Parameters

pVal—The returned string, indicating the user name for Kerberos authentication.
newVal—The set string, indicating the user name for Kerberos authentication.

Basic Example

```
Dim Transport As HETransport

Dim strUserName As String

Set Transport = CreateObject("HETransport.HETransport")



'get kerberos user name

strUserName = Transport.KerberosUsername



'set kerberos user name

strUserName = "ValidName"

Transport.KerberosUsername = strUserName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL, CLSCTX_INPROC_SERVER,
IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrKerberosUserName = NULL;



//get kerberos user name

m_pTransport->get_KerberosUsername(&bstrKerberosUserName);



//set kerberos user name

bstrKerberosUserName = SysAllocString(OLESTR("ValidName”));

m_pTransport->put_KerberosUsername(bstrKerberosUserName);



SysFreeString(bstrKerberosUserName);
```

Property: IHETransport::KerberosVersion 3270 VT

This property returns or sets the Kerberos version to use for the current session. By default, this property is set to 4.

Basic Syntax

Integer = HETransport.**KerberosVersion**
HETransport.**KerberosVersion** = Integer

C++ Syntax

HRESULT IHETransport::**get_KerberosVersion**([out, retval] short * *pVal*);
HRESULT IHETransport::**put_KerberosVersion**([in] short * *newVal*);

Parameters

pVal—The returned value, indicating the Kerberos version of the current session.
newVal—The set value, indicating the Kerberos version of the current session.

Basic Example

```
Dim Transport As HETransport

Dim iVersion As Integer

Set Transport = CreateObject("HETransport.HETransport")



'get kerberos version

iVersion = Transport.KerberosVersion



'set kerberos version

iVersion = 0

Transport.KerberosVersion = iVersion
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

short iKerberosVersion;



//get kerberos version

m_pTransport->get_KerberosVersion(&iKerberosVersion);



//set kerberos version

iKerberosVersion = 0

m_pTransport->put_KerberosVersion(iKerberosVersion);
```

Property: IHETransport::Keyboard 5250

This property returns or sets a value that specifies the type of keyboard being used.

Basic Syntax

String = HETransport.Keyboard
HETransport.Keyboard = String

C++ Syntax

HRESULT IHETransport::**get_Keyboard**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_Keyboard**([in] BSTR *newVal*);

Parameters

pVal—The returned value, specifying the keyboard type.
newVal—The set value, specifying the keyboard type.

Basic Example

```
Dim Transport As HETransport

Dim strKeyboard As String

Set Transport = CreateObject("HETransport.HETransport")

'get keyboard

strKeyboard = Transport.Keyboard

'set keyboard

strKeyboard = "ValidValue"

Transport.Keyboard = strKeyboard
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrKeyboard = NULL;



// get keyboard

m_pTransport->get_Keyboard(&bstrKeyboard);



//set keyboard

bstrKeyboard = SysAllocString(OLESTR("ValidValue”));

m_pTransport->put_ Keyboard(bstrKeyboard);



SysFreeString(bstrKeyboard);
```

Property: IHETransport::LockOnAttention 3270 5250 VT

This property returns or sets a value that indicates whether the keyboard is locked after you send an attention key command. By default, this property is set to VARIANT_FALSE.

Basic Syntax

Boolean = HETransport.**LockOnAttention**

HETransport.**LockOnAttention** = Boolean

C++ Syntax

HRESULT IHETransport::**get_LockOnAttention**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHETransport::**put_LockOnAttention**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the keyboard is locked. A returned value of VARIANT_FALSE indicates that the keyboard is not locked.

newVal—A value of VARIANT_TRUE indicates that the keyboard is locked. A value of VARIANT_FALSE indicates that the keyboard is not locked.

Basic Example

```
Dim Transport As HETransport

Dim bStatus As Boolean

Set Transport = CreateObject("HETransport.HETransport")

'get the lock status

bStatus = Transport.LockOnAttention

'set the lock status

bStatus = False

Transport.LockOnAttention = bStatus
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bLock;



//get lock on attention status

m_pTransport->get_LockOnAttention(&bLock);



//set lock on attention status

bLock = BOOL_FALSE;

m_pTransport->put_LockOnAttention(bLock);
```

Property: IHETransport::MaxRecvBlockSize 3270 5250 VT

This property returns or sets the maximum Winsock receive block size.

Basic Syntax

Long = HETransport.**MaxRecvBlockSize**
HETransport. **MaxRecvBlockSize** = Long

C++ Syntax

HRESULT IHETransport::**get_MaxRecvBlockSize**([out, retval] long * *pVal*);
HRESULT IHETransport::**put_MaxRecvBlockSize**([in] long *newVal*);

Parameters

pVal—The returned value, indicating the maximum Winsock receive block size.
newVal—The set value, indicating the maximum Winsock receive block size.

Basic Example

```
Dim Transport As HETransport

Dim BlockSize As Long



Set Transport =
CreateObject("HETransport.HETransport")



BlockSize = 0

BlockSize = Transport.MaxRecvBlockSize
```

C++ Example

```
IHETransport* pTransport;

long BlockSize = 0;



HRESULT hr = HECoCreateInstance(CLSID_HETransport,

IID_IHETransport,

(LPVOID*)&pTransport,

"HETransport");



if (SUCCEEDED(hr))

{

pTransport ->
get_MaxRecvBlockSize(&BlockSize);

}
```

Property: IHETransport::MessageQueueLibrary 5250

This property returns or sets a value that specifies the library containing the queue object that directs the print output.

Basic Syntax

String = HETransport.**MessageQueueLibrary**
HETransport.**MessageQueueLibrary** = String

C++ Syntax

HRESULT IHETransport::**get_MessageQueueLibrary**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_MessageQueueLibrary**([in] BSTR *newVal*);

Parameters

pVal—The returned value, indicating the library that contains the queue object.
newVal—The set value, indicating the library that contains the queue object.

Basic Example

```
Dim Transport As HETransport

Dim strLibrary As String

Set Transport = CreateObject("HETransport.HETransport")

'get the library

strLibrary = Transport.MessageQueueLibrary

'set the library

strLibrary = "ValidValue"

Transport.MessageQueueLibrary = strLibrary
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrLibrary = NULL;



//get message queue library

m_pTransport->get_MessageQueueLibrary(&bstrLibrary);



//set message queue library

bstrLibrary = SysAllocString(OLESTR("ValidValue”));

m_pTransport->put_MessageQueueLibrary(bstrLibrary);



SysFreeString(bstrLibrary);
```

Property: IHETransport::MessageQueueName 5250

This property returns or sets a value that specifies the queue object that directs the print output.

Basic Syntax

String = HETransport.**MessageQueueName**
HETransport.**MessageQueueName** = String

C++ Syntax

HRESULT IHETransport::**get_MessageQueueName**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_MessageQueueName**([in] BSTR *newVal*);

Parameters

pVal—The returned value, indicating the queue object.
newVal—The set value, indicating the queue object.

Basic Example

```
Dim Transport As HETransport

Dim strName As String

Set Transport = CreateObject("HETransport.HETransport")

'get the name

strName = Transport.MessageQueueName

'set the name

strName = "ValidName"

Transport.MessageQueueName = strName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrName = NULL;



//get message queue name

m_pTransport->get_MessageQueueName(&bstrName);



//set message queue name

bstrName = SysAllocString(OLESTR("ValidName”));

m_pTransport->put_MessageQueueName(bstrName);



SysFreeString(bstrName);
```

Property: IHETransport::ModelColumns 3270 5250 VT

This property returns or sets the number of columns required for the terminal. By default, this property is set to 80 columns.

Basic Syntax

Integer = HETransport.**ModelColumns**
HETransport.**ModelColumns** = Integer

C++ Syntax

HRESULT IHETransport::**get_ModelColumns**([out, retval] short * *pVal*);
HRESULT IHETransport::**put_ModelColumns**([in] short *newVal*);

Parameters

pVal—The returned number of columns required for the terminal.
newVal—The set number of columns required for the terminal.

Basic Example

```
Dim Transport As HETransport

Dim iColumns As Integer

Set Transport = CreateObject("HETransport.HETransport")

'get model column

iColumns = Transport.ModelColumns

'set model column

iColumns = 35

Transport.ModelColumns = iColumns
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

short iModelCol;



//get model column number

m_pTransport->get_ModelColumns(&iModelCol);



//set model column number

iModelCol = 35;

m_pTransport->put_ModelColumns(iModelCol);
```

Property: IHETransport::ModelRows 3270 5250 VT

This property returns or sets the number of rows required for the terminal. By default, this property is set to 24 rows.

Basic Syntax

Integer = HETransport.**ModelRows**
HETransport.**ModelRows** = Integer

C++ Syntax

HRESULT IHETransport::**get_ModelRows**([out, retval] short * *pVal*);
HRESULT IHETransport::**put_ModelRows**([in] short *newVal*);

Parameters

pVal—The returned number of rows required for the terminal.
newVal—The set number of rows required for the terminal.

Basic Example

```
Dim Transport As HETransport

Dim iRows As Integer

Set Transport = CreateObject("HETransport.HETransport")

'get model rows

iRows = Transport.ModelRows

'set model rows

iRows = 80

Transport.ModelRows = iRows
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

short iModelRows;



//get model row number

m_pTransport->get_ModelRows(&iModelRows);



//set model row number

iModelRows = 80;

m_pTransport->put_ModelRows(iModelRows);
```

Property: IHETransport::Password 5250

This property returns or sets a string that specifies the password used to connect to the host.

Basic Syntax

String = HETransport.**Password**
HETransport.**Password** = String

C++ Syntax

HRESULT IHETransport::**get_Password**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_Password**([in] BSTR *newVal*);

Parameters

pVal—The returned string which specifies the password.
newVal—The set string which specifies the password.

Basic Example

```
Dim Transport As HETransport

Dim strPassword As String

Set Transport = CreateObject("HETransport.HETransport")

'get password

strPassword = Transport.Password

'set password

strPassword = "ValidPassword"

Transport.Password = strPassword
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrPassW = NULL;



//get password

m_pTransport->get_Password(&bstrPassW);



//set password

bstrPassW = SysAllocString(OLESTR("ValidValue”));

m_pTransport->put_Password(bstrPassW);



SysFreeString(bstrPassW);
```

Property: IHETransport::PerformingTransfer VT

This property returns or sets a value that indicates whether the Transport object is in file transfer mode.

Basic Syntax

Boolean = HETransport.**PerformingTransfer**

HETransport.**PerformingTransfer** = Boolean

C++ Syntax

HRESULT IHETransport::**get_PerformingTransfer**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IHETransport::**put_PerformingTransfer**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the Transport object is in file transfer mode. A returned value of VARIANT_FALSE indicates that the Transport object is not in file transfer mode.

newVal—A value of VARIANT_TRUE indicates that the Transport object is in file transfer mode. A value of VARIANT_FALSE indicates that the Transport object is not in file transfer mode.

Basic Example

```
Dim Transport As HETransport

Dim bStatus As Boolean

Set Transport = CreateObject("HETransport.HETransport")

'get transfer status

bStatus = Transport.PerformingTransfer

'set transfer status

bStatus = True

Transport.PerformingTransfer = bStatus
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bPerform;



//get transfer status

m_pTransport->get_PerformingTransfer(&bPerform);



//set transfer status

bPerform = VARIANT_TRUE;

m_pTransport->put_PerformingTransfer(bPerform);
```

Property: IHETransport::Port 3270 5250 VT

This property returns or sets a value that specifies the Internet port to which you are connected. You can select a number between 1 and 65534. By default, this option is set to 23.

Basic Syntax

Long = HETransport.**Port**
HETransport.**Port** = Long

C++ Syntax

HRESULT IHETransport::**get_Port**([out, retval] long * *pVal*);
HRESULT IHETransport::**put_Port**([in] long *newVal*);

Parameters

pVal—The returned value, specifying the Internet port.
newVal—The set value, specifying the Internet port.

Basic Example

```
Dim Transport As HETransport

Dim lPort As Long

Set Transport = CreateObject("HETransport.HETransport")



'get port

lPort = Transport.Port



'set port

lPort = 1

Transport.Port = 1
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

Long PortNum;



// get port

PortNum = m_pTransport->get_Port();



//set port

IHETransport * m_pTransport;

Long PortNum = 1;

m_pTransport->put_Port(PortNum);
```

Property: IHETransport::SecurityOption 3270 5250 VT

This property returns or sets a value that specifies the traffic for securing the channel between the server and the client. By default, this option is set to HOSTEX_SECURITY_NO_SECURITY.

Basic Syntax

HOSTEX_SECURITY_OPTIONS = HETransport.**SecurityOption**

HETransport.**SecurityOption** = HOSTEX_SECURITY_OPTIONS

C++ Syntax

HRESULT IHETransport::**get_SecurityOption**([out, retval] HOSTEX_SECURITY_OPTIONS * *pVal*);

HRESULT IHETransport::**put_SecurityOption**([in] HOSTEX_SECURITY_OPTIONS *newVal*);

Parameters

pVal—The returned value, specifying the security option.

newVal—The set value, specifying the security method.

Basic Example

```
Dim Transport As HETransport

Dim SecVal As HOSTEX_SECURITY_OPTIONS



Set Transport =
CreateObject("HETransport.HETransport")



SecVal = HOSTEC_SECURITY_KERBEROS

Transport.SecurityOption = SecVal
```

C++ Example

```
IHETransport* pTransport;

HOSTEX_SECURITY_OPTIONS SecurityOption;



HRESULT hr = HECoCreateInstance(CLSID_HETransport,

IID_IHETransport,

(LPVOID*)&pTransport,

"HETransport");



SecurityOption = HOSTEX_SECURITY_KERBEROS



if (SUCCEEDED(hr))

{

pTransport ->
put_SecurityOption(&SecurityOption);

}
```

Related Topics

HOSTEX_SECURITY_OPTIONS Data Type

Property: IHETransport::SessionKeepAlive 3270 5250 VT

This property returns or sets a value that indicates the time interval (in seconds) before the Transport object sends Keep Alive messages to the host. By default, this property is set to 30 seconds.

Basic Syntax

Long = HETransport.SessionKeepAlive

HETransport.SessionKeepAlive = Long

C++ Syntax

HRESULT IHETransport::**get_SessionKeepAlive**([out, retval] long * *pVal*);

HRESULT IHETransport::**put_SessionKeepAlive**([in] long *newVal*);

Parameters

pVal—The returned value, indicating the time interval before the Transport object sends Keep Alive messages to the host.

newVal—The set value, indicating the time interval before the Transport object sends Keep Alive messages to the host.

Basic Example

```
Dim Transport As HETransport

Dim lSessionAlive As Long

Set Transport = CreateObject("HETransport.HETransport")

'get the session value

lSessionAlive = Transport.SessionKeepAlive

'set the session value

lSessionAlive = 10

Transport.SessionKeepAlive = lSessionAlive
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

long lSessionAlive;



//get the session value

m_pTransport->get_SessionKeepAlive(&lSessionAlive);



//set the session value

lSessionAlive = 10;

m_pTransport->put_SessionKeepAlive(lSessionAlive);
```

Property: IHETransport::TelnetEcho 3270 5250 VT

This property returns or sets a value that indicates how HostExplorer responds to remote echo negotiation with a Telnet host. By default, this property is set to HOSTEX_TELNETECHO_AUTOMATIC.

Basic Syntax

HOSTEX_TELNETECHO = HETransport.**TelnetEcho**

HETransport.**TelnetEcho** = HOSTEX_TELNETECHO

C++ Syntax

HRESULT IHETransport::**get_TelnetEcho**([out, retval] HOSTEX_TELNETECHO * *pVal*);

HRESULT IHETransport::**put_TelnetEcho**([in] HOSTEX_TELNETECHO *newVal*);

Parameters

pVal—The returned value, indicating how HostExplorer responds to remote echo negotiation with a Telnet host.

newVal—The set value, indicating how HostExplorer responds to remote echo negotiation with a Telnet host.

Basic Example

```
Dim Transport As HETransport

Dim EchoValue As HOSTEX_TELNETECHO

Set Transport = CreateObject("HETransport.HETransport")

'get telnet echo

EchoValue = Transport.TelnetEcho

'set telnet echo

EchoValue = HOSTEX_TELNETECHO_YES

Transport.TelnetEcho = EchoValue
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

HOSTEX_TELNETECHO telnetEcho;



//get telnet echo

m_pTransport->get_TelnetEcho(&telnetEcho);



//set telnet echo

telnetEcho = HOSTEX_TELNETECHO_YES;

m_pTransport->put_TelnetEcho(telnetEcho);
```

Related Topics

HOSTEX_TELNETECHO Data Type

Property: IHETransport::TelnetIsLineMode 3270 5250 VT

This property returns a value that indicates whether HostExplorer stores characters in a buffer until you send a carriage return to the host. When enabled, Linemode forces HostExplorer to send characters one line at a time rather than as individual characters.

Basic Syntax Boolean = HETransport.**TelnetIsLineMode**

C++ Syntax HRESULT IHETransport::**get_TelnetIsLineMode**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that you are in Linemode. A returned value of VARIANT_FALSE indicates that you are not in Linemode.

Basic Example

```
Dim Transport As HETransport

Dim bLineMode As Boolean

Set Transport = CreateObject("HETransport.HETransport")

bLineMode = Transport.TelnetIsLineMode
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bLineMode;

m_pTransport->get_TelnetIsLineMode(&bLineMode);
```

Property: IHETransport::TelnetIsLocalEcho 3270 5250 VT

This property returns a value that indicates whether you are currently in local echo mode.

Basic Syntax Boolean = HETransport.**TelnetIsLocalEcho**

C++ Syntax HRESULT IHETransport::**get_TelnetIsLocalEcho**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that you are in local echo mode. A returned value of VARIANT_FALSE indicates that you are not in local echo mode.

Basic Example

```
Dim Transport As HETransport

Dim bLocalEcho As Boolean

Set Transport = CreateObject("HETransport.HETransport")

bLocalEcho = Transport.TelnetIsLocalEcho
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bLocalEcho;

m_pTransport->get_TelnetIsLocalEcho(&bLocalEcho);
```

Property: IHETransport::TelnetName 3270 5250 VT

This property returns or sets a string that specifies a name to override the name used during Telnet negotiation with the host.

Basic Syntax

```
String = HETransport.TelnetName
HETransport.TelnetName = String
```

C++ Syntax

```
HRESULT IHETransport::get_TelnetName([out, retval] BSTR * pVal);
HRESULT IHETransport::put_TelnetName([in] BSTR newVal);
```

Parameters

pVal—The returned string, specifying a Telnet name.
newVal—The set string, specifying a Telnet name.

Basic Example

```
Dim Transport As HETransport

Dim strName As String

Set Transport = CreateObject("HETransport.HETransport")

'get telnet name

strName = Transport.TelnetName

'set telnet name

strName = "ValidTelnetName"

Transport.TelnetName = strName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrName = NULL;
```

```
//get telnet name

m_pTransport->get_TelnetName(&bstrName);



//set telnet name

bstrName = SysAllocString(OLESTR("ValidName”));

m_pTransport->put_ TelnetName(bstrName);



SysFreeString(bstrName);
```

Property: IHETransport::TerminalModel 3270 5250 VT

This property returns or sets a value that indicates the type of terminal that you are using to connect to the host. For TN3270 and TN5250 terminals, the default for this property is set to HOSTEX_TERM_MODEL_2. For TNVT terminals, the default is set to HOSTEX_TERM_MODEL_VT220.

Basic Syntax

HOSTEX_TERM_MODEL = HETransport.**TerminalModel**

HETransport.**TerminalModel** = HOSTEX_TERM_MODEL

C++ Syntax

HRESULT IHETransport::**get_TerminalModel**([out, retval] HOSTEX_TERM_MODEL * *pVal*);

HRESULT IHETransport::**put_TerminalModel**([in] HOSTEX_TERM_MODEL *newVal*);

Parameters

pVal—The returned value, indicating the terminal type.

newVal—The set value, indicating the terminal type.

Basic Example

```
Dim Transport As HETransport

Dim Model As HOSTEX_TERM_MODEL

Set Transport = CreateObject("HETransport.HETransport")



'get terminal model

Model = Transport.TerminalModel



'set terminal model

Model = HOSTEX_TERM_MODEL_3

Transport.TerminalModel = Model
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

HOSTEX_TERM_MODEL Model;



//get terminal model

m_pTransport->get_TerminalModel(&Model);



//set terminal model

Model = HOSTEX_TERM_MODEL_3;

m_pTransport->put_TerminalModel(Model);
```

Related Topics

HOSTEX_TERM_MODEL Data Type

Property: IHETransport::TNESession 3270

This property returns or sets a value that indicates whether the terminal is in TN3270E mode.

Basic Syntax

Boolean = HETransport.**TNESession**

C++ Syntax

HRESULT IHETransport::**get_TNESession**([out, retval] VARIANT_BOOL * *pVal*);

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the terminal is in TN3270E mode. A returned value of VARIANT_FALSE indicates that the terminal is not in TN3270E mode.

Basic Example

```
Dim Transport As HETransport

Dim bTNESession As Boolean

Set Transport = CreateObject("HETransport.HETransport")

bTNESession = Transport.TNESession
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

VARIANT_BOOL bTNESession;

m_pTransport->get_TNESession(&bTNESession);
```

Property: IHETransport::TraceFilename 3270 5250 VT

This property returns or sets a string that specifies the name of the file that contains the information that is sent to and received from the host. By default, this property is set to C:\HETRACE.TRC.

Basic Syntax

String = HETransport.**TraceFilename**
HETransport.**TraceFilename** = String

C++ Syntax

HRESULT IHETransport::**get_TraceFilename**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_TraceFilename**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying the file name.
newVal—The set string, specifying the file name.

Basic Example

```
Dim Transport As HETransport

Dim strName As String

Set Transport = CreateObject("HETransport.HETransport")



'get trace file name

strName = Transport.TraceFilename



'set trace file name

strName = "ValidFileName"

Transport.TraceFilename = strName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrFile = NULL;



//get trace file name

m_pTransport->get_TraceFilename(&bstrFile);



//set trace file name

bstrFile = SysAllocString(OLESTR("ValidFileName”));

m_pTransport->put_TraceFilename(bstrFile);



SysFreeString(bstrFile);
```

Property: IHETransport::Username 5250

This property returns or sets a string that specifies the user name that you use to connect to the host.

Basic Syntax

String = HETransport.**Username**
HETransport.**Username** = String

C++ Syntax

HRESULT IHETransport::**get_Username**([out, retval] BSTR * *pVal*);
HRESULT IHETransport::**put_Username**([in] BSTR *newVal*);

Parameters

pVal—The returned string, specifying your user name.
newVal—The set string, specifying your user name.

Basic Example

```
Dim Transport As HETransport

Dim strName As String

Set Transport = CreateObject("HETransport.HETransport")



'get user name

strName = Transport.Username

'set user name

strName = "ValidUserName"

Transport.Username = strName
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

BSTR bstrName = NULL;



//get user name

m_pTransport->get_Username(&bstrName);



//set user name

bstrName = SysAllocString(OLESTR("ValidUserName”));

m_pTransport->put_Username(bstrName);



SysFreeString(bstrName);
```

Property: IHETransport::VTLineMode VT

This property returns or sets a value that indicates whether or not you are currently in Linemode. By default, this property is set to HOSTEX_LINEMODE_DONTDOLINEMODE.

Basic Syntax

HOSTEX_LINEMODE = HETransport.**VTLineMode**

HETransport.**VTLineMode** = HOSTEX_LINEMODE

C++ Syntax

HRESULT IHETransport::**get_VTLineMode**([out, retval] HOSTEX_LINEMODE * *pVal*);

HRESULT IHETransport::**put_VTLineMode**([in] HOSTEX_LINEMODE *newVal*);

Parameters

pVal—A returned value of TRUE indicates that you are currently in Linemode. A returned value of FALSE indicates you are not currently in Linemode.

newVal—A value of TRUE indicates that you are currently in Linemode. A value of FALSE indicates you are not currently in Linemode.

Basic Example

```
Dim Transport As HETransport

Dim LineMode As HOSTEX_LINEMODE

Set Transport = CreateObject("HETransport.HETransport")

'get VT line mode

LineMode = Transport.VTLineMode

'set VT line mode

LineMode = HOSTEX_LINEMODE_ALWAYS

Transport.VTLineMode = LineMode
```

C++ Example

```
IHETransport * m_pTransport = NULL;

HRESULT hr = CoCreateInstance(CLSID_HETransport, NULL,
CLSCTX_INPROC_SERVER, IID_IHETransport, (LPVOID *)&m_pTransport);

if (FAILED(hr))

{

// add code

return S_FALSE;

}

HOSTEX_LINEMODE LineMode;



//get VT line mode

m_pTransport->get_VTLineMode(&LineMode);



//set VT line mode

LineMode = HOSTEX_LINEMODE_ALWAYS;

m_pTransport->put_VTLineMode(LineMode);
```

Related Topics

HOSTEX_LINEMODE Data Type

Data Types of the Transport Objects

The Transport objects contain the following data types:

HOSTEX_ATN_FORMAT

HOSTEX_CON_STATUS

HOSTEX_DEVICE_TYPE

HOSTEX_ENCRYPTED

HOSTEX_FUNCTION_KEY

HOSTEX_LINEMODE

HOSTEX_SECURITY_OPTIONS

HOSTEX_TELNETECHO

HOSTEX_TERM_MODEL

HOSTEX_TOGGLE_RECEIVE

Related Topics

Methods of the Transport Objects

Properties of the Transport Objects

HOSTEX_LINEMODE Data Type VT

The HOSTEX_LINEMODE data type specifies how HostExplorer stores characters in a buffer until you send a carriage return to the host. When enabled, Line mode forces HostExplorer to send characters one line at a time rather than as individual characters. Using line mode is useful when you are trying to reduce costs on networks that charge per packet or when you are experiencing long network delays. By default, this data type is set to HOSTEX_LINEMODE_DONTDOLINEMODE.

It has the following values:

HOSTEX_LINEMODE_DONTDOLINEMODE	Disables Line mode.
HOSTEX_LINEMODE_ALWAYS	Enables Line mode continuously.
HOSTEX_LINEMODE_DURINGLOCALECHO	Enables Line mode when the host tells HostExplorer to do the echoing.
HOSTEX_LINEMODE_WHENNOTINSGA	Enables Line mode when the host does not Suppress Go Ahead (SGA).
HOSTEX_LINEMODE_LOCALECHOORNOTSGA	Enables Line mode when the host tells HostExplorer to do the echoing or when the host does not SGA.

Related Topics

Property: IHESession::VTLineMode

HOSTEX_TELNETECHO Data Type VT

The HOSTEX_TELNETECHO data type specifies how HostExplorer responds to remote echo negotiation with a Telnet host. By default, this data type is set to HOSTEX_TELNETECHO_AUTOMATIC.

It has the following values:

HOSTEX_TELNETECHO_NO	Indicates that HostExplorer negotiates remote echo with the host without local echoing.
HOSTEX_TELNETECHO_YES	Indicates that HostExplorer negotiates local echo with the host and always echoes.
HOSTEX_TELNETECHO_AUTOMATIC	Indicates that HostExplorer uses host commands to negotiate remote or local echoing.

Related Topics

Property: IHETransport::TelnetEcho

Property: IHESession::TelnetEcho

HOSTEX_CON_STATUS Data Type 3270 5250 VT

The HOSTEX_CON_STATUS data type specifies the current status of your connection to the host.

It has the following values:

HOSTEX_CON_STATUS_DISCONNECTED	Indicates when you are fully disconnected from the host.
HOSTEX_CON_STATUS_CONNECTED	Indicates when you are fully connected to the host.
HOSTEX_CON_STATUS_CONNECTING	Indicates the status from the time you issue a Connect command to the host to the time you are actually connected to the host.
HOSTEX_CON_STATUS_DISCONNECTING	Indicates the status from the time you issue a Disconnect command to the host to the time you are fully disconnected from the host.

Related Topics

Property: IHETransport::ConnectionStatus

HOSTEX_DEVICE_TYPE Data Type 3270 5250 VT

The HOSTEX_DEVICE_TYPE data type specifies the device type to be used with the Transport objects.

It has the following values:

HOSTEX_DEVICE_TYPE_DISPLAY	Indicates that the Transport objects are used with a display terminal.
HOSTEX_DEVICE_TYPE_PRINTER	Indicates that the Transport objects are used with a printer device.

Related Topics

Property: IHETransport::DeviceType

HOSTEX_FUNCTION_KEY Data Type 3270 5250 VT

The HOSTEX_FUNCTION_KEY data type specifies the value that you can request the Transport object to send to the host.

It has the following values:

HOSTEX_FUNCTION_KEY_SYSTEM_ REQUEST	Executes a system-request command to the host.
HOSTEX_FUNCTION_KEY_SEND_ ATTENTION	Executes an attention command to the host.
HOSTEX_FUNCTION_KEY_SEND_ ABORT_OUTPUT	Executes an abort-output command to the host; this command stops the process.

Related Topics

Method: IHETransport::SendFunctionKey

HOSTEX_TERM_MODEL Data Type 3270 5250 VT

The HOSTEX_TERM_MODEL data type specifies the type of terminal that you are using to connect to the host. By default, this data type is set to HOSTEX_TERM_MODEL_2.

It has the following values:

HOSTEX_TERM_MODEL_2	Indicates that the terminal consists of 24 lines by 80 columns.
HOSTEX_TERM_MODEL_3	Indicates that the terminal consists of 32 lines by 80 columns.
HOSTEX_TERM_MODEL_4	Indicates that the terminal consists of 27 lines by 80 columns.
HOSTEX_TERM_MODEL_5	Indicates that the terminal consists of 27 lines by 132 columns.
HOSTEX_TERM_MODEL_VT100	Indicates that you are using a VT100 terminal to connect to the host.
HOSTEX_TERM_MODEL_VT101	Indicates that you are using a VT101 terminal to connect to the host.
HOSTEX_TERM_MODEL_VT102	Indicates that you are using a VT102 terminal to connect to the host.
HOSTEX_TERM_MODEL_VT220	Indicates that you are using a VT220 terminal to connect to the host.
HOSTEX_TERM_MODEL_VT320	Indicates that you are using a VT320 terminal to connect to the host.
HOSTEX_TERM_MODEL_VT420	Indicates that you are using a VT420 terminal to connect to the host.
HOSTEX_TERM_MODEL_ANSI	Indicates that you are using an ANSI/BBS terminal to connect to the host.
HOSTEX_TERM_MODEL_SCOANSI	Indicates that you are using a SCO-ANSI terminal to connect to the host.
HOSTEX_TERM_MODEL_TERM_ IBM3151	Indicates that you are using an IBM 3151 terminal to connect to the host.

Related Topics

Property: IHETransport::TerminalModel

HOSTEX_TOGGLE_RECEIVE Data Type 3270 5250 VT

The HOSTEX_TOGGLE_RECEIVE data type specifies how HostExplorer toggles the state of the receipt of data from the Transport objects.

It has the following values:

HOSTEX_TOGGLE_RECEIVE_RETURNS_ STATE	Indicates that HostExplorer returns only the actual state (whether or not the receipt is already blocked).
HOSTEX_TOGGLE_RECEIVE_STATE	Indicates that HostExplorer toggles the current state. If the state is TRUE, it is toggled to FALSE. If the state is FALSE, it is toggled to TRUE.
HOSTEX_TOGGLE_RECEIVE_OFF	Indicates that if the state is ON, HostExplorer toggles it to OFF.
HOSTEX_TOGGLE_RECEIVE_ON	Indicates that if the state is OFF, HostExplorer toggles it to ON.

Related Topics

Method: IHETransport::ToggleBlockReceive

TN5250 Language-Conversion Table

The following list consists of the available languages and the associated language index for TN5250 terminals. This list is a subset of the contents in the language-conversion table. Refer to Hex5250.hxl, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the Hex5250.hxl file is available. 

5250 Languages	Language Index
Albania-MNCS	0x0142
Austrian/German-273	0x0100
Austrian/German-1141	0x0101
Austrian/German-MNCS	0x0102
Belgian-MNCS	0x0103
Belgian-1148	0x0103
Bosnian/Herzegovinian-870	0x014C
Brazilian Portuguese-037	0x0105
Brazilian Portuguese-1140	0x0141
Bulgarian-1025	0x010A
Canadian French-MNCS	0x0106
Canadian French-037	0x0107
Canadian French-1140	0x0108
Croatian-870	0x0109
Cyrillic-880	0x010B
Czech-870	0x010C
Danish-277	0x010D
Danish-1142	0x010E
Danish-MNCS	0x0143
Dutch-037	0x010F
Dutch-1140	0x0144
Dutch-MNCS	0x0110
UK English-285	0x0111
UK English-1146	0x0112
UK English-MNCS	0x0145
US English-037	0x0113
US English-1140	0x014D
US English-MNCS	0x0146
Estonian-1122	0x0115
Finnish-278	0x0116
Finnish-MNCS	0x0114
Finnish-1143	0x0117
French (Azerty)-297	0x0118
French (Azerty)-1147	0x0119
French (Azerty)-MNCS	0x011A
French (Qwerty)-297	0x0147
French (Qwerty)-1147	0x0148
French (Qwerty)-MNCS	0x0149
Greek New-875	0x011F
Hungarian-870	0x0120
Icelandic-1149	0x0121
Icelandic-871	0x0122
Icelandic-MNCS	0x011B
Italian-280	0x0123
Italian-1144	0x0124
Italian-MNCS	0x0125
Latin-2-870	0x0126
Latvian-1112	0x0127
Lithuanian-1112	0x0128
FYR Macedonia	0x011C
Norwegian-277	0x012B
Norwegian-1142	0x012C
Norwegian-MNCS	0x011D

Polish-870	0x012D
Portuguese-037	0x012E
Portuguese-MNCS	0x012F
Portuguese-1140	0x0130
Romanian-870	0x0131
Russian-1025	0x0133
Serbian Cyrillic-1025	0x0134
Serbian/Montenegro Latin-870	0x0129
Slovenian-870	0x0135
Slovakian-870	0x012A
Spanish-284	0x0136
Spanish-MNCS	0x012A
Spanish-1145	0x0132
Spanish Speaking-284	0x0137
Spanish Speaking-1145	0x0138
Spanish Speaking-MNCS	0x014A
Swedish-278	0x0139
Swedish-1143	0x013A
Swedish-MNCS	0x014B
Swiss French-MNCS	0x013B
Swiss French-1148	0x013C
Swiss German-MNCS	0x013D
Swiss German-1148	0x013E
Turkish-1026	0x013F
Ukrainian-1123	0x0140

Sample of Hex5250.hxl

The following is a sample of Hex5250.hxl, the language-conversion table for TN5250 terminals:

[US English-037]	The language string as it appears in the Translation Table Options dialog box under the Terminal interface.
HostCodepage=037	The host code page.
AnsiCodePage=1252	The ANSI code page.
KBDID5250=USB	The host keyboard ID.
CSID5250=00697	The host character set ID.
CPID5250=00037	The host code page ID.
INIFILE=037.HXL	The .hxl file containing the EBCDIC-to-ASCII and EBCDIC-to-Unicode conversion tables.
IndexCharset=0x0113	The language index for the conversion tables.

HOSTEX_SECURITY_OPTIONS Data Type

The HOSTEX_SECURITY_OPTIONS data type specifies the type of security method used to secure the traffic between the server and the client. By default, this data type is set to HOSTEX_SECURITY_NO_SECURITY.

It has the following values:

HOSTEX_SECURITY_NO_SECURITY	Indicates that there is no security of traffic between the server and the client.
HOSTEX_SECURITY_SSL_TLS	Encrypts all traffic between the server and the client for 3270, 5250, and Telnet terminals using Secure Socket Layer.
HOSTEX_SECURITY_KERBEROS	Provides authentication and encrypts all traffic betweent the server and the client for 3270 and Telnet terminals using Kerberos software.
HOSTEX_SECURITY_SECURESHELL	Encrypts all traffic between the server and the client for Telnet terminals using Secure Shell software.

Related Topics

Property: IHETransport::SecurityOption

Property: IHESessionSecurity::SecurityOption

HOSTEX_ENCRYPTED Data Type 3270 5250 VT

The HOSTEX_ENCRYPTED data type specifies the encryption level of the session.

It has the following values:

HOSTEX_IS_NOT_ENCRYPTED	Indicates that the session is not encrypted.
HOSTEX_IS_PARTIALLY_ENCRYPTED	Indicates that the session is partially encrypted.
HOSTEX_IS_ENCRYPTED	Indicates that the session is fully encrypted.

Related Topics

Property: IHETransport::IsEncrypted

TNVT UPSS Language-Conversion Table

The following list consists of the available UPSS (User Preferred Supplemental Character Set) and the associated language index for TNVT terminals. This list is a subset of the contents in the language-conversion table. Refer to VT_UPS.hxl, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the VT_UPS.hxl file is available. 

VT UPSS Languages	Language Index
DEC Special	0x0030
DEC Supplemental	0x005B
DEC Technical	0x003E
Europa3 (333)	0x0007
DEC Greek (373)	0x0008
ISO Latin-1 (8859-1)	0x007B
ISO Latin-2 (8859-2)	0x0002
ISO Latin-4 (8859-4)	0x000B
ISO Cyrillic (8859-5)	0x000C
ISO Greek (8859-7)	0x000D
ISO Latin-5 (8859-9)	0x0006
ISO Latin 9 (8859-15)	0x0009
PC English (437)	0x0001
PC Icelandic (861)	0x000E
PC Modern Greek (869)	0x000F
PC Modern Turkish (857)	0x0010
PC Multilingual (850)	0x0011
PC Nordic (865)	0x0012
PC Portuguese (860)	0x0013
PC Slavic (852)	0x0014
PC Canadian-French (863)	0x0015
PC Cyrillic (855)	0x0016
PC Cyrillic (866)	0x0017
PC Baltic (921)	0x0018
PC Estonian (922)	0x0019
PC Greek (851)	0x001A
HP Roman 8 (1051)	0x001B
PC Greek (437G, 210)	0x001C
HP Roman 9	0x001D
ISO Latin-3 (8859-3)	0x001E
PC Baltic (775)	0x0021
Windows Latin-2 (1250)	0x0022
Windows Cyrillic (1251)	0x0023
Windows Latin-1 (1252)	0x0024
Windows Greek (1253)	0x0025
Windows Latin-5 Turkish (1254)	0x0026
Windows Baltic (1257)	0x0027
ISO Latin-6 (8859-10)	0x0028
ISO Latin-7 (8859-13)	0x0029
ISO Latin-8 (8859-14)	0x002A
IBM3151 (Special Graphics)	0x002B
Slovenian 7-Bit	0x002C

Sample of VT_UPS.hxl

The following is a sample of VT_UPS.hxl, the UPSS language-conversion table for TNVT terminals:

[PC English (437)]	The language string as it appears in the VT Character Set dialog box under the Terminal interface.
DEFINE=VTCS_PCENGLISH437	An internal string.
INIFILE=437.HXL	The .hxl file that contains the host-to-ASCII and host-to-Unicode conversion tables.
IndexCharset=0x0001	The language index for the conversion tables.

HOSTEX_GRAPHICS_CURSOR_TYPE Data Type 3270

The HOSTEX_GRAPHICS_CURSOR_TYPE data type specifies how the cursor appears in the terminal window. By default, this data type is set to HOSTEX_GRAPHICS_CURSOR_TYPE_SMALL_CROSS_WHITE.

It has the following values:

HOSTEX_GRAPHICS_CURSOR_TYPE_ SMALL_CROSS_WHITE	Displays the cursor as a small white cross.
HOSTEX_GRAPHICS_CURSOR_TYPE_ LARGE_CROSS_WHITE	Displays the cursor as a large white cross.
HOSTEX_GRAPHICS_CURSOR_TYPE_ SMALL_CROSS_GREEN	Displays the cursor as a small green cross.
HOSTEX_GRAPHICS_CURSOR_TYPE_ LARGE_CROSS_GREEN	Displays the cursor as a large green cross.

Related Topics

Property: IHEParser::GraphicsCursorType

Property: IHESessionGraphics::GraphicsCursorType

HOSTEX_GRAPHICS_MODEL Data Type 3270

The HOSTEX_GRAPHICS_MODEL data type sets general graphic options. By default, this data type is set to HOSTEX_GRAPHICS_MODEL_3270PCG.

It has the following values:

HOSTEX_GRAPHICS_MODEL_ NOGRAPHICS	Indicates that HostExplorer displays only text.
HOSTEX_GRAPHICS_MODEL_3179G	Indicates that HostExplorer displays the IBM 3179G graphics terminal model.
HOSTEX_GRAPHICS_MODEL_3472G	Indicates that HostExplorer displays the IBM 3472G graphics terminal model.
HOSTEX_GRAPHICS_MODEL_3270PCG	Indicates that HostExplorer displays the IBM 3270G graphics terminal model.

Related Topics

Property: IHEParser::GraphicsModel

Property: IHESessionGrahics::GraphicsModel

HOSTEX_KEYBOARD_TYPE Data Type 3270 5250 VT

The HOSTEX_KEYBOARD_TYPE data type specifies the type of keyboard to use for the current session.

It has the following values:

HOSTEX_KEYBOARD_TYPE_PC_84	Indicates that the PC keyboard has 84 keys.
HOSTEX_KEYBOARD_TYPE_PC_101	Indicates that the PC keyboard has 101 keys.
HOSTEX_KEYBOARD_TYPE_PC_102	Indicates that the PC keyboard has 102 keys.
HOSTEX_KEYBOARD_TYPE_DEC_LK450	Indicates that the DEC keyboard is an LK450 model.
HOSTEX_KEYBOARD_TYPE_IBM_3270	Indicates that the IBM keyboard is a 3270 model.
HOSTEX_KEYBOARD_TYPE_PC_104	Indicates that the PC keyboard has 104 keys.

Related Topics

Property: IHESessionKeyboard::KeyboardType

TNVT NRC Language-Conversion Table

The following list consists of the available NRC (National Replacement Character) support languages and the associated language index for TNVT terminals. This list is a subset of the contents in the language-conversion table. Refer to VT_NRC.hxl, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the VT_NRC.hxl file is available. 

VT NRC Support Languages	Language Index
None	0x0000
DEC Swedish	0x0037
DEC French Canadian	0x0039
DEC Swiss	0x003D
ISO United Kingdom	0x0041
DEC Finnish	0x0043
DEC Norwegian/Danish	0x0045
ISO German	0x004B
ISO French	0x0052
ISO Italian	0x0059
DEC Portuguese	0x005D
ISO Spanish	0x005A
ISO Norwegian/Danish	0x0060

Sample of VT_NRC.hxl

The following is a sample of VT_NRC.hxl, the NRC language-conversion table for TNVT terminals:

[ISO German]	The language string as it appears in the VT Character Set dialog box under the Terminal interface.
IndexCharset=0x004B	The language index for the conversion tables.
Hex23=0x23	The new byte character that replaces the original value at the same character position.
Hex23U=0x0023	The new Unicode character that replaces the original value at the same character position.

HOSTEX_CELL_DELIMITED Data Type 3270 5250

The HOSTEX_CELL_DELIMITED data type specifies how HostExplorer parses screen data when copying data to the Clipboard in cell-delimited format.

It has the following values:

HOSTEX_CELL_DELIMITED_WORD	Indicates that HostExplorer parses screen data at words.
HOSTEX_CELL_DELIMITED_FIELD	Indicates that HostExplorer parses screen data at field attributes.

Related Topics

Property: IHEParser::CellDelimited

Property: IHESessionEditing::CellDelimited

HOSTEX_STATUS_LINE_MODE Data Type 3270 5250

The HOSTEX_STATUS_LINE_MODE data type indicates where the status line appears.

It has the following values:

HOSTEX_STATUS_LINE_MODE_ NOSTATUSLINE	Indicates that no status line is displayed.
HOSTEX_STATUS_LINE_MODE_ TERMINALSTATUSLINE	Indicates that the status line appears at the bottom of the terminal screen.
HOSTEX_STATUS_LINE_MODE_ WINDOWSTATUSBAR	Indicates that the status line appears at the bottom of the window.

Related Topics

Property: IHEParser::StatusLineMode

Property: IHESessionDisplay::StatusLineMode

OLE_COLOR Data Type 3270 5250 VT

The OLE_COLOR data type is used for properties that return colors. When a property is declared as OLE_COLOR, the Properties window displays a color palette, which you can use to select the color for the property rather than using the numeric equivalent.

This data type consists of a long value that is converted from a hexadecimal value. The hexadecimal value returns the corresponding values for BGR (Blue, Green, Red). The first two hexadecimal numbers refer to Blue; the middle two numbers refer to Green, and the last two numbers refer to Red. For example:

· Blue—hexadecimal FF0000; long 16711680

· Green—hexadecimal FF00; long 65280

· Red—hexadecimal FF; long 255

Note: A hexadecimal value must be converted to a long value so that you can use it in a property.

RelatedTopics

Property: IHESessionDisplay::WorkspaceBackgroundColor

Property: IHESessionDisplay::WorkspaceForegroundColor

HOSTEX_SWITCHSCREENTYPE Data Type 3270 5250 VT

The HOSTEX_SWITCHSCREENTYPE data type specifies the type of information that HostExplorer retains when the host switches the screen between standard and alternate sizes. By default, this data type is set to HOSTEX_SWITCHSCREENTYPE_KEEPSIZE.

It has the following values:

HOSTEX_SWITCHSCREENTYPE_ KEEPSIZE	Indicates that when HostExplorer switches between screen sizes, if the current font is not available, HostExplorer selects another font within given parameters.
HOSTEX_SWITCHSCREENTYPE_ KEEPFONT	Indicates that when HostExplorer switches between screen sizes, it keeps the font size constant.
HOSTEX_SWITCHSCREENTYPE_ KEEPOLDINFO	Indicates that when HostExplorer switches between screen sizes, it saves the font and window information separately for the default and alternate modes.

Related Topics

Property: IHESessionFonts::SwitchScreenType

HOSTEX_CAPTURE_MODE Data Type VT

The HOSTEX_CAPTURE_MODE data type specifies how to capture selected text.

It has the following values:

HOSTEX_CAPTURE_MODE_RAW	Indicates that the system captures all data, including escape sequences, received by the emulator.
HOSTEX_CAPTURE_MODE_BINARY	Indicates that escape sequences are removed so that what appears on the screen is what is sent to the printer. In this mode, the system captures every line that is terminated by a line feed, thereby allowing you to capture line-by-line output.

Related Topics

Property: IHEParser::CaptureMode

HOSTEX_CUT_MODE Data Type 3270 5250

The HOSTEX_CUT_MODE data type indicates what occurs on the screen after you cut text. By default, this data type is set to HOSTEX_CUT_MODE_DELETE_TEXT.

It has the following values:

HOSTEX_CUT_MODE_REPLACE_WITH_ SPACES	Indicates that the cut text is replaced with blank characters.
HOSTEX_CUT_MODE_REPLACE_WITH_ NULLS	Indicates that the cut text is replaced with nulls (or zeros).
HOSTEX_CUT_MODE_DELETE_TEXT	Indicates that the cut text is deleted and not replaced by any characters.

Related Topics

Property: IHEParser::Cutmode

HOSTEX_FIELD_ATTR_REPLACEMENT Data Type 3270 5250

The HOSTEX_FIELD_ATTR_REPLACEMENT data type specifies how HostExplorer replaces the field attribute when copying information to the clipboard. By default, this data type is set to HOSTEX_FIELD_ATTR_REPLACEMENT_COMMA.

It has the following values:

HOSTEX_FIELD_ATTR_REPLACEMENT_NONE	Indicates that HostExplorer does not replace the field attribute with anything.
HOSTEX_FIELD_ATTR_REPLACEMENT_TAB	Indicates that HostExplorer replaces the field attribute with a tab stop on the screen.
HOSTEX_FIELD_ATTR_REPLACEMENT_ COMMA	Indicates that HostExplorer replaces the field attribute with a comma on the screen.
HOSTEX_FIELD_ATTR_REPLACEMENT_ PARAGRAPH	Indicates that HostExplorer replaces the field attribute with a paragraph mark on the screen.

HOSTEX_INSERT_KEY_STYLE Data Type 3270

The HOSTEX_INSERT_KEY_STYLE data type specifies how the Insert key option operates.

It has the following values:

HOSTEX_INSERT_KEY_STYLE_RESET	Indicates that the Insert-key option is on until you press the Reset key.
HOSTEX_INSERT_KEY_STYLE_ACTION	Indicates that the Insert-key option is on until you press an action key, such as Enter or Clear.

Related Topics

Property: IHEParser::InsertKeyStyle

HOSTEX_NEXT_FIELD_KEY Data Type 3270 5250

The HOSTEX_NEXT_FIELD_KEY data type specifies how HostExplorer tabs to the next field on the screen. By default, this data type is set to HOSTEX_NEXT_FIELD_KEY_COMMA. You can set this data type only when the HOSTEX_PASTE_MODE data type is set to HOSTEX_PASTE_MODE_PASTE_FIELD.

It has the following values:

HOSTEX_NEXT_FIELD_KEY_NONE	Indicates that HostExplorer does not interpret any of the characters as the next field key.
HOSTEX_NEXT_FIELD_KEY_TAB	Indicates that HostExplorer tabs to the next field using the Tab character.
HOSTEX_NEXT_FIELD_KEY_COMMA	Indicates that HostExplorer tabs to the next field using the Comma character.
HOSTEX_NEXT_FIELD_KEY_PARAGRAPH	Indicates that HostExplorer tabs to the next field using the Paragraph Mark or Enter key.

Related Topics

Property: IHEParser::NextFieldKey

HOSTEX_PASTE_MODE Data Type

HOSTEX_PASTE_MODE Data Type 3270 5250

The HOSTEX_PASTE_MODE data type specifies how HostExplorer pastes the contents of the Clipboard to the current cursor location. By default, this data type is set to HOSTEX_PASTE_MODE_PASTE_BLOCK.

It has the following values:

HOSTEX_PASTE_MODE_PASTE_BLOCK	Indicates that HostExplorer stops pasting text when it reaches a protected field on the screen.
HOSTEX_PASTE_MODE_PASTE_OVERLAY	Indicates that HostExplorer ignores pasted characters that overlay protected fields.
HOSTEX_PASTE_MODE_PASTE_STREAM	Indicates that HostExplorer pastes text one character at a time and stops when it reaches a protected field.
HOSTEX_PASTE_MODE_PASTE_ STREAMWORDWRAP	Indicates that HostExplorer pastes text using wordwrap. In this case, HostExplorer pastes text, stops at a protected field, and continues pasting at the next available unprotected field.
HOSTEX_PASTE_MODE_PASTE_FIELD	Indicates that HostExplorer pastes text in a stream-like fashion, and moves to the next field when a HOSTEX_NEXT_FIELD_KEY character is encountered.

Related Topics

Property: IHEParser::PasteMode

HOSTEX_NEXT_FIELD_KEY Data Type

HOSTEX_SELECTION_MODE Data Type 3270 5250 VT

The HOSTEX_SELECTION_MODE data type specifies how you select text. By default, this data type is set to HOSTEX_SELECTION_MODE_BLOCK.

It has the following values:

HOSTEX_SELECTION_MODE_BLOCK	Indicates that you can select text as a block.
HOSTEX_SELECTION_MODE_STREAM	Indicates that you can select text as a stream.

Related Topics

Property: IHEParser::SelectionMode

HOSTEX_TERMINAL_ID Data Type VT

The HOSTEX_TERMINAL_ID data type specifies the terminal ID response that HostExplorer sends to the host.

It has the following values:

HOSTEX_TERMINAL_ID_VT100	Indicates that the terminal ID response is VT100.
HOSTEX_TERMINAL_ID_VT101	Indicates that the terminal ID response is VT101.
HOSTEX_TERMINAL_ID_VT102	Indicates that the terminal ID response is VT102.
HOSTEX_TERMINAL_ID_VT220	Indicates that the terminal ID response is VT220.
HOSTEX_TERMINAL_ID_VT320	Indicates that the terminal ID response is VT320.
HOSTEX_TERMINAL_ID_VT420	Indicates that the terminal ID response is VT420.
HOSTEX_TERMINAL_ID_VT80	Indicates that the terminal ID response is VT80.
HOSTEX_TERMINAL_ID_VT100J	Indicates that the terminal ID response is VT100J.
HOSTEX_TERMINAL_ID_VT102J	Indicates that the terminal ID response is VT102J.
HOSTEX_TERMINAL_ID_VT220J	Indicates that the terminal ID response is VT220J.
HOSTEX_TERMINAL_ID_VT282	Indicates that the terminal ID response is VT282.
HOSTEX_TERMINAL_ID_VT382	Indicates that the terminal ID response is VT382.

Related Topics

Property: IHEParser::TerminalID

HOSTEX_TRANSFER_HOSTTYPE Data Type 3270

The HOSTEX_TRANSFER_HOSTTYPE data type specifies the operating system run by the host. By default, this data type is set to HOSTEX_TRANSFER_HOSTTYPE_CMS.

It has the following values:

HOSTEX_TRANSFER_HOSTTYPE_ CMS	Indicates that you are uploading files to a host that is running CMS.
HOSTEX_TRANSFER_HOSTTYPE_ TSO_ MUSIC	Indicates that you are uploading files to a host that is running TSO/MUSIC.
HOSTEX_TRANSFER_HOSTTYPE_ CICS	Indicates that you are uploading files to a host that is running CICS.

HOSTEX_TRANSFER_INITIALACTION Data Type 3270

The HOSTEX_TRANSFER_INITIALACTION data type specifies the action that HostExplorer performs before uploading or downloading files. By default, this data type is set to HOSTEX_TRANSFER_INITIALACTION_NONE.

It has the following values:

HOSTEX_TRANSFER_INITIALACTION_ NONE	Indicates that HostExplorer is not to perform an action before transferring a file.
HOSTEX_TRANSFER_INITIALACTION_ HOMEKEY	Indicates that HostExplorer is to send a Home-key command to the host before transferring a file.
HOSTEX_TRANSFER_INITIALACTION_ ENTERKEY	Indicates that HostExplorer is to send an Enter key or carriage return (CR)to the host before transferring a file.

HOSTEX_TRANSFER_RECORDFORMAT Data Type 3270

The HOSTEX_TRANSFER_RECORDFORMAT data type specifies general record formats.

It has the following values:

HOSTEX_TRANSFER_RECORDFORMAT_ DEFAULT	Indicates that the host portion of the file transfer must use the default record format.
HOSTEX_TRANSFER_RECORDFORMAT_ FIXED	Indicates that HostExplorer must upload the file to a fixed-record-format file.
HOSTEX_TRANSFER_RECORDFORMAT_ VARIABLE	Indicates that HostExplorer must upload the file to a variable-record-format file.
HOSTEX_TRANSFER_RECORDFORMAT_ UNDEFINED	Indicates that the record format is undefined; it applies only to Multiple Virtual Storage (MVS).

HOSTEX_TRANSFER_TARGET Data Type 3270 VT

The HOSTEX_TRANSFER_TARGET data type specifies where the file transfer occurs.

It has the following values:

HOSTEX_TRANSFER_PC_FILESYSTEM	Indicates that the file transfer occurs on the file system.
HOSTEX_TRANSFER_PC_CLIPBOARD	Indicates that the file transfer occurs on the Clipboard.

HOSTEX_TRANSFER_TYPE Data Type 3270 VT

The HOSTEX_TRANSFER_TYPE data type specifies the name of the host file transfer program to use when uploading and/or downloading files.

It has the following values:

HOSTEX_TRANSFER_TYPE_INDFILE	Indicates that the name of the file-transfer program for TN3270 terminals is IND$File. You can use IND$File to send and receive TN3270 files.
HOSTEX_TRANSFER_TYPE_ZMODEM	Indicates that the name of the file-transfer program is Zmodem. This VT protocol provides end-to-end data security between program applications while significantly eliminating file-transfer errors.
HOSTEX_TRANSFER_TYPE_YMODEM	Indicates that the name of the file-transfer program is Ymodem. This VT protocol supports batch file transfers and can send the file name and file size before the actual data.
HOSTEX_TRANSFER_TYPE_XMODEM	Indicates that the name of the file-transfer program is Xmodem. This VT protocol is one of the original protocols written for asynchronous communications.
HOSTEX_TRANSFER_TYPE_XMODEM1K	Indicates that the name of the file-transfer program is Xmodem-1K. This VT protocol is part of Xmodem.
HOSTEX_TRANSFER_TYPE_KERMIT	Indicates that the name of the file-transfer program is Kermit. This VT protocol sends batch files with the name and time stamp of each file in small packet sizes. These packets contain fields that mark the beginning, length, type, and sequence number of the packet.

HOSTOVERWRITE_BEHAVIOUR Data Type VT

The HOSTOVERWRITE_BEHAVIOUR data type specifies how files are written to the host.

It has the following values:

OVERWRITE_BEHAVIOUR_ NEWERORLONGER	Indicates that HostExplorer overwrites the existing file only if the file you are sending is newer or longer.
OVERWRITE_BEHAVIOUR_APPEND	Indicates that HostExplorer adds the file your are sending to the end of the existing file.
OVERWRITE_BEHAVIOUR_ALWAYS	Indicates that HostExplorer always overwrites the existing file.
OVERWRITE_BEHAVIOUR_ FILESIZEDATEDIFFER	Indicates that HostExplorer overwrites the existing file only if the file you are sending is a different size and/or is newer.
OVERWRITE_BEHAVIOUR_NEVER	Indicates that HostExplorer never overwrites the existing file.
OVERWRITE_BEHAVIOUR_NEWER	Indicates that HostExplorer overwrites the existing file only if the file you are sending is newer.

HEPARSER_FEATURE Data Type

The HEPARSER_FEATURE data type indicates the type of feature that is enabled or disabled.

It has the following values:

HOSTEX_8BIT_MODE	Determines whether the communication mode used to connect to the host supports the 8-bit data format. This data type value applies only to TNVT terminals.
HOSTEX_ALA_ENABLED	Determines whether HostExplorer enables ALA (A Program Language) support. By default, this data type value is FALSE, and applies only to TN3270 terminals.
HOSTEX_ALA_KEY_TABLE	Determines whether HostExplorer uses the ALA keys definition. By default, this data type value is FALSE, and applies only to TN3270 terminals.
HOSTEX_ALLOW_AID_KEY_REPEAT	Determines whether HostExplorer sends multiple function-key commands to the host without you having to press a key again. By default, this data type value is FALSE.
HOSTEX_ALWAYS_OUTLINE	Determines whether HostExplorer automatically sets all input fields to full outline. By default, this data type value is FALSE and applies only to TN3270 terminals.
HOSTEX_AUTO_COPY_KEEP_SELECTION	Determines whether HostExplorer maintains the selection once you have copied or cut the text. By default, this data type value is FALSE.
HOSTEX_AUTO_COPY_SELECTED_TEXT	Determines whether HostExplorer automatically copies all selected text to the Clipboard. By default, this data type value is FALSE.
HOSTEX_AUTO_DIACRITIC_ COMPOSITION	Determines whether HostExplorer can compose accented and/or special characters. By default, this data type value is FALSE.
HOSTEX_BACKSPACE_IS_DELETE	Determines whether HostExplorer sends a Delete character to the host when you press the Backspace key. This data type value applies only to TNVT terminals.
HOSTEX_CAN_CHANGE_SCREEN	Determines whether you can update the screen. This data type value is set by code processing.
HOSTEX_CAPTURE_SCREEN	Determines whether HostExplorer saves all information on the screen. By default, this data type value is FALSE.
HOSTEX_CLIPBOARD_FORMAT_BITMAP	Determines whether HostExplorer enables bitmap format when copying data to the Clipboard. By default, this data type value is TRUE.
HOSTEX_CLIPBOARD_FORMAT_CSV	Determines whether HostExplorer enables CSV format when copying data to the Clipboard and pasting data from other applications. By default, this data type value is TRUE.

HOSTEX_CLIPBOARD_FORMAT_HE	Determines whether HostExplorer enables its proprietary format when copying data to the Clipboard. By default, this data type value is TRUE.
HOSTEX_CLIPBOARD_FORMAT_PASTE_ LINK	Determines whether HostExplorer enables Paste Link format when copying data to the Clipboard. By default, this data type value is TRUE.
HOSTEX_CLIPBOARD_FORMAT_RTF	Determines whether HostExplorer enables Rich Text Format (RTF) when copying data to the Clipboard. By default, this data type value is TRUE.
HOSTEX_CLIPBOARD_FORMAT_TEXT	Determines whether HostExplorer enables standard text format when copying data to the Clipboard and pasting data from other applications. By default, this data type value is TRUE.
HOSTEX_CONTROLCODES	Determines whether HostExplorer acts on control codes or displays them using a special character set. This data type value is set by the host and applies only to TNVT terminals.
HOSTEX_CURSOR_KEY_MODE	Determines the cursor-key mode (Normal or Application), which affects the sequences HostExplorer sends to the host. This data type value applies only to TNVT terminals.
HOSTEX_ENABLE_NOTIFY	Determines whether HostExplorer beeps and the session window is not the active or highlighted window. By default, this data type value is FALSE.
HOSTEX_ENABLE_SOUND	Determines whether HostExplorer emits all program sounds. By default, this data type value is FALSE.
HOSTEX_ENTRY_ASSIST	Determines whether HostExplorer enables Entry Assist, which lets you set general editing options. By default, this data type value is FALSE and applies only to TN3270 and TN5250 terminals.
HOSTEX_FORCE_ALT_SIZE	Determines whether you can change the window to the alternate size when the host receives an Erase Write command. By default, this data type value is FALSE and applies only to TN3270 terminals.
HOSTEX_HOST_WRITABLE_STATUS_LINE	Determines whether the host displays messages within the status line. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_IGNORE_ATTRIBUTE	Determines whether the field attribute is displayed. By default, this data type value is FALSE.
HOSTEX_INSERT_MODE	Determines whether the Insert key inserts characters in the current and subsequent lines. By default, this data type value is FALSE and applies only to TN3270 and TN5250 terminals.

HOSTEX_ISO_COLORS	Determines whether HostExplorer enables support for ISO colors for ANSI color escape sequences when using VT100, VT101, VT220, VT320, and VT420 models. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_KEYPAD_APPLICATION_MODE	Determines whether HostExplorer sends application sequences to the host. This data type value applies only to TNVT terminals.
HOSTEX_MONO_TRANSITION_MODE	Determines whether the screen is in monochrome transition mode. If it is, the screen is black, and the characters are green. This data type value is set by the host and applies only to TN3270 terminals.
HOSTEX_MOVE_CURSOR_AFTER_PASTE	Determines whether HostExplorer automatically repositions the cursor after pasting text. By default, this data type is FALSE and applies only to TN3270 and TN5250 terminals.
HOSTEX_NEW_LINE_MODE	Determines whether pressing the Enter key sends a carriage-return (CR) or carriage return/line feed (CR/LF) command to the host. By default, the CR option is on. This data type value applies only to TNVT terminals.
HOSTEX_NO_LOCK_KEYBOARD	Determines whether HostExplorer sends a Never Lock the Keyboard command to the host. By default, this data type value is TRUE and applies only to TN3270 terminals.
HOSTEX_PAR_LOCAL_ECHO	Determines whether HostExplorer enables local echo of characters typed in the emulator. This data type value applies only to TNVT terminals.
HOSTEX_PAR_VTONLINE	Determines whether you can type, and move the cursor around the screen without sending data to the host. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_PS_RESERVE	Lets you reserve a session to prevent user input. By default, this data type value is FALSE.
HOSTEX_SAVE_ATTR_IN_SCROLLBACK	Determines whether HostExplorer saves the Telnet screen attributes within data in the Scrollback buffer. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_SAVE_ERASE_SCREENS	Determines whether HostExplorer saves a screen to the Scrollback buffer before performing the Erase-Screen Host command. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_SCROLL_NO_BLANKS	Determines whether HostExplorer prevents adding blank lines to the Scrollback buffer. By default, this data type value is TRUE and applies only to TNVT terminals.

HOSTEX_SMOOTH_SCROLL	Determines whether HostExplorer scrolls data using a smooth scroll method. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_TYPE_AHEAD	Determines whether you can continue typing even when the keyboard is locked. HostExplorer enables you to continue typing by buffering typed characters. By default, this data type value is FALSE and applies only to TN3270 and TN5250 terminals.
HOSTEX_UNICODE_FONT	Determines whether you are using Unicode font. By default, this data type value is TRUE.
HOSTEX_UNICODE_MODE	Determines whether the Translate object was loaded in the Parser object. By default, this data type value is FALSE.
HOSTEX_VT_ENABLE_BREAK	Determines whether you can send the Break key to the host. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_VT_NRC_MODE	Determines whether HostExplorer enables the NRC (National Replacement Character) set. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_WHAT_THIS	Verifies whether “What’s this?” or context-sensitive Help is enabled. By default, this data type value is FALSE.
HOSTEX_WORD_WRAP	Determines whether HostExplorer automatically wraps text around the screen. Text wrapping occurs when the terminal attempts to display a character beyond the last column of the emulator. By default, this data type value is FALSE and applies only to TNVT terminals.

TN3270 Language-Conversion Table

The following list consists of the available languages and the associated language index for TN3270 terminals. This list is a subset of the contents in the language-conversion table. Refer to Hex3270.hxl, a file that is shipped with the HostExplorer product, for the complete table.

A sample of the Hex3270.hxl file is available. 

3270 Languages	Language Index
Austrian ECECP (1141)	0x0200
Austrian CECP (273)	0x0201
Austrian (273)	0x0202
Belgian ECECP (1148)	0x0203
Belgian CECP (500)	0x0204
Belgian (500)	0x0205
Brazilian (275)	0x0206
Canadian Bilingual ECECP (1140)	0x0207
Canadian Bilingual CECP (037)	0x0208
Canadian Bilingual (037)	0x0209
Croatian	0x020A
Cyrillic (880)	0x0241
Cyrillic (1025)	0x020B
Czech (870)	0x020C
Danish ECECP (1142)	0x020D
Danish CECP (277)	0x020E
Danish (277)	0x020F
Dutch CECP (037)	0x0245
Dutch ECECP (1140)	0x024F
English-UK ECECP (1146)	0x0210
English-UK CECP (285)	0x0211
English-UK (285)	0x0212
English-US C370 CECP V2 (1047)	0x0213
English-US C370 CECP (037)	0x0214
English-US CECP (037)	0x0215
English-US (037)	0x0216
Estonian (1122)	0x024B
Finnish ECECP (1143)	0x0217
Finnish CECP (278)	0x0218
Finnish (278)	0x0219
French ECECP (1147)	0x021A
French CECP (297)	0x021B
French (297)	0x021C
German ECECP (1141)	0x0248
German CECP (273)	0x0249
German (273)	0x024A
Greek (423)	0x0240
Greek New (875)	0x021D
Hungarian (870)	0x021E
Icelandic ECECP (1149)	0x023E
Icelandic CECP (871)	0x023F
Italian ECECP (1144)	0x0220
Italian (280)	0x0221
Latin-2 (870)	0x0222
Latin-9 (924)	0x0250
Latvian (1112)	0x024C
Lithuanian (1112)	0x024D
Netherlands ECECP (1140)	0x0224
Netherlands CECP (037)	0x0225
Netherlands (037)	0x0226
Norwegian CECP (1142)	0x0227
Norwegian CECP (277)	0x0228
Norwegian (277)	0x0229

Polish (870)	0x022A
Portuguese ECECP (1140)	0x022B
Portuguese CECP (037)	0x022C
Portuguese (037)	0x022D
ROECE Latin/Yugoslav (870)	0x022E
Romanian (870)	0x022F
Russian (1025)	0x0247
Serbian (870)	0x0243
Slovenian (870)	0x0244
Spanish Speaking ECECP (1145)	0x0231
Spanish Speaking CECP (284)	0x0232
Spanish Speaking (284)	0x0233
Spanish CECP (284)	0x0234
Swedish ECECP (1143)	0x0235
Swedish CECP (278)	0x0236
Swedish (278)	0x0237
Swiss French ECECP (1148)	0x0238
Swiss French CECP (500)	0x0239
Swiss French (500)	0x023A
Swiss German ECECP (1148)	0x023B
Swiss German CECP (500)	0x023C
Swiss German (500)	0x023D
Turkish (905)	0x0242
Turkish New (1026)	0x0223
Ukrainian (1123)	0x024E

Sample of Hex3270.hxl

The following is a sample of Hex3270.hxl, the language-conversion table for TN3270 terminals:

[English-US C370 CECP V2 (1047)]	The language string as it appears in the Translation Table Options dialog box under the Terminal interface.
HostCodepage=037	The host code page.
AnsiCodePage=1252	The ANSI code page.
acCECP0=0x00	The first CECP byte ID.
acCECP1=0x65	The second CECP byte ID.
acCECP2=0x00	The third CECP byte ID.
acCECP3=0x25	The fourth CECP byte ID.
INIFILE=037.HXL	The .hxl file containing the EBCDIC-to-ASCII and EBCDIC-to-Unicode conversion tables.
IndexCharset=0x0216	The language index for the conversion tables.

HOSTEX_TPRINT_OUTPUT Data Type 3270

The HOSTEX_TPRINT_OUTPUT data type specifies where a host TPRINT or PCPRINT print job is being sent.

It has the following values:

HOSTEX_TPRINT_OUTPUT_DEFAULT_ WIN_PRINTER	Indicates that the print job is being sent to the default printer specified on your machine.
HOSTEX_TPRINT_OUTPUT_LPT1	Indicates that the print job is being sent to the LPT1 port.
HOSTEX_TPRINT_OUTPUT_LTP2	Indicates that the print job is being sent to the LPT2 port.
HOSTEX_TPRINT_OUTPUT_LPT3	Indicates that the print job is being sent to the LPT3 port.
HOSTEX_TPRINT_OUTPUT_CLIPBOARD	Indicates that the print job is being sent to the Clipboard.

Related Topics

Property: IHEParser::TPrintOutput

HOSTEX_TRANSFER Data Type 3270

The HOSTEX_TRANSFER data type specifies whether to download or upload files.

It has the following values:

HOSTEX_TRANSFER_DOWNLOAD	Indicates that HostExplorer downloads files.
HOSTEX_TRANSFER_UPLOAD	Indicates that HostExplorer uploads files.

Related Topics

Property: IHEParser::XferMode

PC_OVERWRITE_BEHAVIOUR Data Type 3270 VT

The PC_OVERWRITE_BEHAVIOUR data type specifies how files are written to the PC.

It has the following values:

OVERWRITE_BEHAVIOUR_OVERWRITE	Indicates that HostExplorer overwrites the existing file with the file you are sending.
OVERWRITE_BEHAVIOUR_RENAME	Indicates that HostExplorer renames the file you are sending and leaves the existing file unchanged.
OVERWRITE_BEHAVIOUR_SKIP	Indicates that HostExplorer transfers all of the files except for those that already exist.

Relationship Between COM Objects

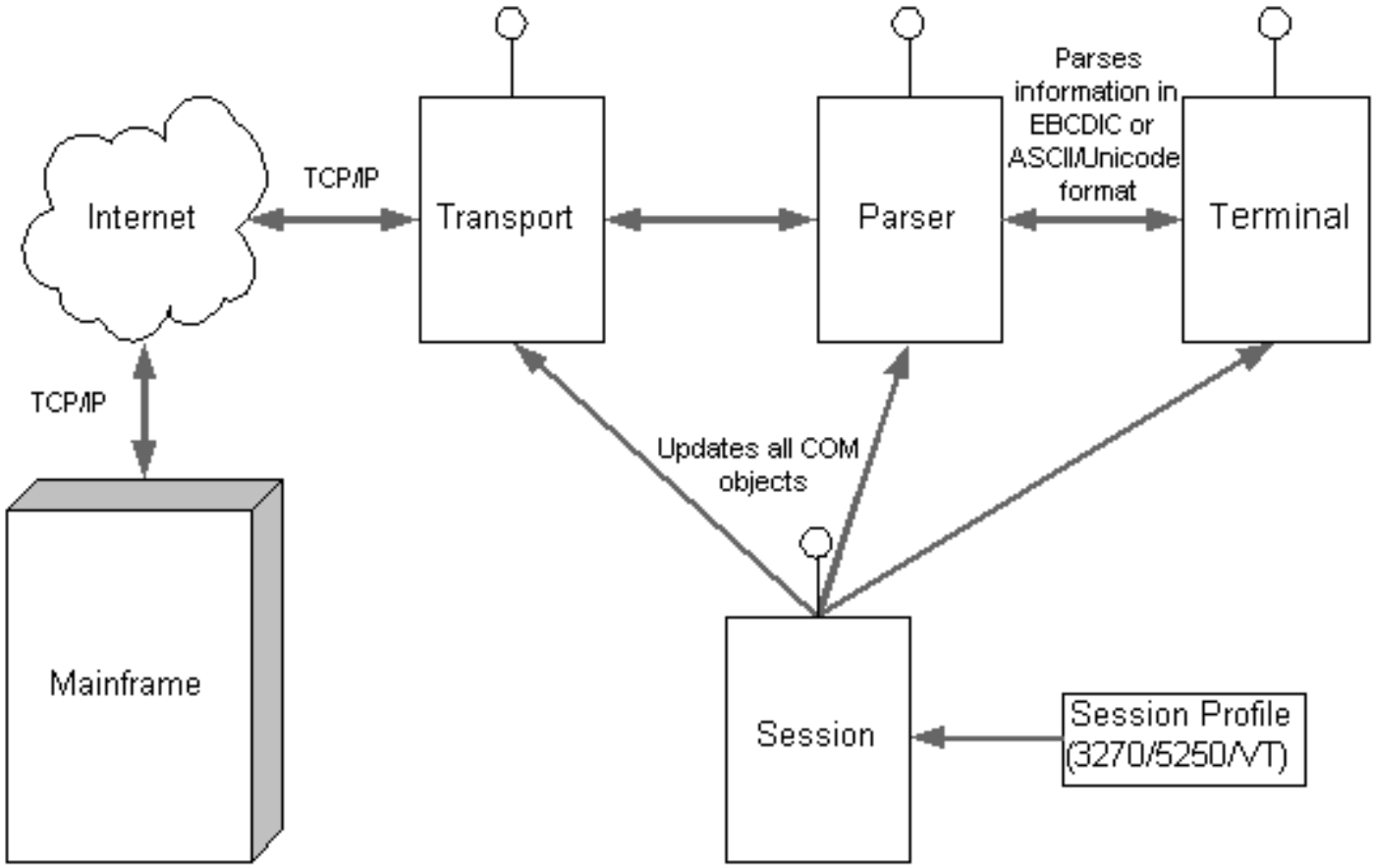
The following diagram illustrates the relationships between the following four COM objects:

· Terminal

· Session

· Parser

· Transport



Related Topics

About COM Objects

About COM Objects

Common Object Model (COM) is an efficient object-oriented programming methodology that documents all of the standard functions that reside in DLLs. COM allows programmers to develop objects that can be accessed by any COM-compliant application.

COM is one type of application programming interface (API), which is a widely known document standard used to write applications. COM is also a new form of Object Linking and Embedding (OLE), a document standard that lets you create objects within one application and embed them in another application.

In COM, an individual object is assigned discrete and logical functionality. As well, you can create relationships between objects. Because objects can be independent of one another, you can create an object that inherits many of its features from existing objects, rather than changing a module when a new object is added. Using small and flexible COM objects, you can:

· improve application performance

· reduce application size

· reuse code to develop applications more rapidly

The COM interface displays the available API methods and properties with the corresponding syntax. The interface executes the code in a dynamic-link library (DLL), and the COM object returns the corresponding value or data.

An ActiveX object (for example, Terminal objects within HostExplorer) is a specific type of COM object and is associated with the GUI. ActiveX objects support a number of standard methods and properties.

Sample files of COM objects are available in the HostExplorer\SDK\Samples\COMObjects directory where the program files are stored on your machine.

Related Topics

About the Terminal Objects

About the Session Object

About the Parser Objects

About the Transport Objects

About OHIO

OHIO (Open Host Interface Objects) is a standardized programming interface to the host data. OHIO provides a common access method to the data when it arrives at the client and divides the data into logical objects. Using OHIO, you can write a script that can run in any type of emulator that supports OHIO.

The HostExplorer object Ohio is a base interface for eight Ohio interfaces, which use a specific inheritance hierarchy.

Ohio is used to create a screen that lets you communicate with, and connect to or disconnect from, the host. Ohio functions are used to call Transport and Parser functions.

Sample files of Ohio are available in the HostExplorer\SDK\Samples\OHIO directory where the program files are stored on your machine.

OhioManager Interface

The OhioManager interface is a central repository that provides a list of all available OhioSession objects. It allows you to open and close sessions.

Methods

The OhioManager interface consists of the following methods:

CloseSession

OpenSession

Properties

The OhioManager interface consists of the following properties:

OhioVersion

Sessions

VendorName

VendorObject

VendorProductVersion

Method: IOhioManager::CloseSession

This method closes an OhioSession object. The OhioSession is considered invalid and is removed from the list of OhioSession objects.

Basic Syntax OhioManager.**CloseSession**(*inSession* As Variant)

C++ Syntax HRESULT IOhioManager::**CloseSession**([in] VARIANT *inSession*);

Parameters *inSession*—The IOhioSession that you want to close.

Basic Example

```
Dim OManager As OhioManager

Dim OSession As OhioSession

. . .

OManager.CloseSession (OSession)

‘close by session name

OManager.CloseSession ("aix")
```

C++ Example

```
IOhioSession* pOhioSession;

IOhioManager* pOhioManager;

VARIANT vr;

. . .

vr.vt = VT_DISPATCH;

pOhioSession->QueryInterface

(IID_IDispatch, (void **)

&vr.pdispVal);

vr.pdispVal->Release();

pOhioManager->CloseSession(vr);
```

Method: IOhioManager::OpenSession

This method creates an OhioSession object.

Basic Syntax

OhioManager.**OpenSession**(*inConfigResource* As String, *inSessionName* As String) As OhioSession

C++ Syntax

HRESULT IOhioManager::**OpenSession**([in] BSTR *inConfigResource*, [in]BSTR *inSessionName*, [out, retval] IOhioSession * * *outSession*);

Parameters

inConfigResource—The Hummingbird profile name.

inSessionName—The name of the IOhioSession object to be created.

outSession—The returned address of the IOhioSession object that is created.

Basic Example

```
Dim OManager As Object

Dim OSession As Object

Dim strProfile, strSessionName As String

. . .

strProfile = "C:\\Aix.hep"

strSessionName = "aix"

Set OManager =

CreateObject("HEOhio.OhioManager")

Set OSession = OManager.OpenSession

(strProfile,strSessionName)
```

C++ Example

```
IOhioManager* pOhioManager = NULL;

. . .

HRESULT hr = CoCreateInstance

(CLSID_OhioManager, NULL,

CLSCTX_INPROC_SERVER, IID_IOhioManager,

reinterpret_cast<void**>

(&pOhioManager));


if (FAILED(hr)){…} //error


BSTR l_bstrHEProfile, l_bstrSessionName;

l_bstrSessionName = SysAllocString

(OLESTR("aix"));

l_bstrHEProfile = SysAllocString

(OLESTR("C:\\Aix.hep"));

hr = pOhioManager->OpenSession

(l_bstrHEProfile, l_bstrSessionName,

&pOhioSession);

if (FAILED(hr)) {…} //error


SysFreeString(l_bstrHEProfile);

SysFreeString(l_bstrSessionName);
```

Property: IOhioManager::OhioVersion

This property returns the Ohio version level of the implementation.

Basic Syntax String = OhioManager.**OhioVersion**

C++ Syntax HRESULT IOhioManager::**get_OhioVersion**([out, retval] BSTR * *pVal*);

Parameters *pVal*—The returned Ohio version level.

Basic Example

```
Dim OManager As OhioManager

Dim strVersion As String

. . .

strVersion = OManager.OhioVersion
```

C++ Example

```
IOhioManager* pOhioManager;

. . .

BSTR bstrVersion;

HRESULT hr = pIOhioManager->

get_OhioVersion(&bstrVersion);

if (FAILED (hr)) {…} //error
```

Property: IOhioManager::Sessions

This property returns an OhioSessions object containing the OhioSession objects available on the system. Once the OhioSessions object is created, the list of objects is static. Use the OhioSessions.Refresh method to update the list.

Basic Syntax

OhioSessions = OhioManager.**Sessions**

C++ Syntax

HRESULT IOhioManager::**get_Sessions**([out, retval] IOhioSessions * * *pVal*);

Parameters

pVal—The returned address of the IOhioSessions object.

Basic Example

```
Dim OManager As OhioManager

Dim OSessions As OhioSessions

Dim OSess1, OSess2 As OhioSession

Set OSessions = OManager.Sessions

‘retrieve OhioSession object by session

‘name

Set OSess = OManager.Sessions(“aix”)

‘retrieve OhioSession object by index

Set OSess2 = OManager.Sessions(2)
```

C++ Example

```
IOhioManager* pOhioManager;

IOhioSessions* pIOhioSessions = NULL;

. . .

HRESULT = pIOhioManager->

get_Sessions(&pIOhioSessions);

if (FAILED(hr)) {…} //error

long iCount;

pIOhioSessions->get_Count(&iCount);
```

Property: IOhioManager::VendorName

This property returns the name of the vendor that is providing the OHIO implementation.

Basic Syntax String = OhioManager.**VendorName**

C++ Syntax HRESULT IOhioManager::**get_VendorName**([out, retval] BSTR * *pVal*);

Parameters *pVal*—The returned name of the vendor.

Basic Example

```
Dim OManager As OhioManager

Dim strVendorName As String

. . .

strVendorName = OManager.VendorName
```

C++ Example

```
IOhioManager* pOhioManager;

BSTR bstrVendorName;

. . .

HRESULT hr = pOhioManager->

get_VendorName(&bstrVendorName);

if {FAILED(hr)) {…} //error

printf("vendor name is %s\n",OLE2A(bstrVendorName));

SysFreeString(bstrVendorName);
```

Property: IOhioManager::VendorObject

This property returns the address of the vendor object that provides non-standard, vendor-specific extensions to the Ohio object.

Basic Syntax Dispatch = OhioManager.**VendorObject**

C++ Syntax HRESULT IOhioManager::**get_VendorObject**([out, retval] IDispatch * * *pVal*);

Parameters *pVal*—There are currently no extensions implemented in the property VendorObject; therefore, this property returns a null pointer.

Property: IOhioManager::VendorProductVersion

This property returns the vendor product version that is providing the OHIO implementation.

Basic Syntax String = OhioManager.**VendorProductVersion**

C++ Syntax HRESULT OhioManager.**get_VendorProductVersion**([out, retval] BSTR * *pVal*);

Parameters *pVal*—The returned vendor product version.

Basic Example

```
Dim OManager As OhioManager

Dim strVendorProdVer As String

. . .

strVendorProdVer =

OManager.VendorProductVersion
```

C++ Example

```
BSTR bstrVendorProdVer;

IOhioManager* pOhioManager;

. . .

pOhioManager->get_VendorProductVersion

(&bstrVendorProdVer);

Printf("Product version is %d\n", bstrVendorProdVer);

SysFreeString(bstrVendorProdVer);
```

OhioSessions Interface

The OhioSessions interface contains a collection of all the available sessions. You can use this interface to request a specific session or determine the number of sessions available. Because the list of sessions is static, you can use a refresh function to get an updated list.

Methods

The OhioSessions interface consists of the following method:

Refresh

Properties

The OhioSessions interface consists of the following properties:

Count

Item

Method: IOhioSessions::Refresh

This method updates the list of sessions with all the available OhioSession objects. Updating does not preserve the indexing of OhioSession objects.

Basic Syntax

```
OhioSessions.Refresh
```

C++ Syntax

```
HRESULT IOhioSessions::Refresh();
```

Parameters

This method has no parameters.

Basic Example

```
'Here we refresh the OhioSessions object

'before retrieving its third OhioSession

'element, if any

Dim OSessions As OhioSessions

Dim OSess As OhioSession

. . .

OSessions.Refresh

'get the OhioSession object at index 3

Set OSess = OSessions(3)

If (OSess Is Nothing) Then
```

‘no session at index

Else

‘session at index

C++ Example

```
//CloseSession decrements the count. If
//we do not call refresh, the previous
//count will be retrieved.
IOhioManager* pIOhioManager;
IOhioSessions* pIOhioSessions
long lCount;
pIOhioManager->CloseSession(vr); pIOhioSessions->Refresh();
HRESULT hr = pIOhioSessions->get_Count
(&lCount);
if (FAILED(hr)) {…} //error
```

Property: IOhioSessions::Count

This property returns the number of OhioSession objects contained in the collection.

Basic Syntax Long = OhioSessions.**Count**

C++ Syntax HRESULT IOhioSessions::**get_Count**([out, retval] long * *pVal*);

Parameters *pVal*—The returned value of the IOhioSession objects contained in the collection.

Basic Example

```
Dim OManager As OhioManager
Dim OSessions As OhioSessions
Dim num As Long
. . .
‘access the count from OhioSessions object
num = OSessions.Count
‘access count directly from OhioManager
num = OManager.OSessions.Count
```

C++ Example

```
long lCount;
IOhioSessions* pIOhioSessions;
. . .
pIOhioSessions->Refresh();
HRESULT hr = pIOhioSessions->get_Count
(&lCount);
if (FAILED (hr)) {…} //error
```

Property: IOhioSessions::Item

This property returns the OhioSession object at the specified index (one-based indexing) or name of the session.

Basic Syntax

OhioSession = OhioSessions.**Item**

OhioSessions.**Item** = VARIANT

C++ Syntax

HRESULT IOhioSessions::**get_Item**([in] VARIANT *inIndexOrKey*, [out, retval] *IOhioSession* * * *pVal*);

Parameters

inIndexOrKey—The index of the session.

IOhioSession—The name of the session.

pVal—The returned address of the IOhioSession object.

Basic Example

```
Dim OSessions As OhioSessions

Dim OManager As OhioManager

Dim OSess As OhioSession

. . .

OSessions.Refresh

‘access OhioSession object by numeric

‘index

Set OSess = OSessions.Item("1")

‘access OhioSession object by session name

Set OSess = OSessions.Item(“aix”)

‘access OhioSession object directly from

‘OhioManager

Set Osess = OManager.Sessions.Item(“1”);
```

C++ Example

```
IOhioSessions* pIOhioSessions;

IOhioSession* pOSess;

. . .

BSTR bstrSessName;

VARIANT vrSessions;

vrSessions.vt=VT_I4;

vrSessions.lVal=1;

pIOhioSessions->get_Item

(vrSessions,&pOSess);

pOSess->get_SessionName(&bstrSessName);

USES_CONVERSION;

char* szSessName = OLE2A(bstrSessName);

printf("Session Name is %s\n",szSessName);

SysFreeString(bstrSessName)
```

OhioSession Interface

The OhioSession interface allows you to connect to or disconnect from the host and access the host through the screen. It can provide you with the following data:

· session type (for example, 3270, 5250, VT)

· session name (for example, Session 1, Session 2)

· session status (for example, connected or disconnected)

Methods

The OhioSession interface consists of the following methods:

Connect

Disconnect

isConnected

OnSessionChanged

Properties

CanChangeScreen

ConfigurationResource

Screen

SessionName

SessionType

Method: IOhioSession::Connect

This method starts the communications link to the host.

Basic Syntax	OhioSession.**Connect**
C++ Syntax	HRESULT IOhioSession::**Connect**();
Parameters	This method has no parameters.
Basic Example	Dim OManager As Object
	Dim OSession As Object
	Dim strProfile, strSessionName As String
	. . .
	strProfile = "C:\\Aix.hep"
	strSessionName = "aix"
	Set OManager = CreateObject
	("HEOhio.OhioManager")
	Set OSession = OManager.OpenSession
	(strProfile, strSessionName)
	OSession.Connect

C++ Example

```
IOhioManager* pIOhioManager;
IOhioSession* pIOhioSession;
. . .
HRESULT hr = CoCreateInstance
(CLSID_OhioManager, NULL,
CLSCTX_INPROC_SERVER, IID_IOhioManager,
reinterpret_cast<void**>
(&pIOhioManager));

if (FAILED(hr)) {…} //error

BSTR bstrHEProfile, bstrSessionName;
bstrSessionName = SysAllocString
(OLESTR("aix"));
bstrHEProfile = SysAllocString
(OLESTR("C:\\Aix.hep"));

pIOhioManager->OpenSession
(l_bstrHEProfile, bstrSessionName,
&pIOhioSession);

SysFreeString(bstrHEProfile);
SysFreeString(bstrSessionName);

pIOhioSession->Connect();
pIOhioSession->Disconnect();
```

Method: IOhioSession::Disconnect

This method stops the communications link to the host.

Basic Syntax OhioSession.**Disconnect**

C++ Syntax HRESULT IOhioSession::**Disconnect**();

Parameters This method has no parameters.

Basic Example

```
Dim OManager As Object
Dim OSession As Object
Dim strProfile, strSessionName As String
. . .
strProfile = "C:\\Aix.hep"
strSessionName = "aix"
Set OManager = CreateObject
("HEOhio.OhioManager")
Set OSession = OManager.OpenSession
(strProfile, strSessionName)
OSession.Connect
OSession.Disconnect
```

C++ Example

```
IOhioSession* pIOhioSession;
IOhioManager* pIOhioManager;
. . .
HRESULT hr = CoCreateInstance
(CLSID_OhioManager, NULL,
CLSCTX_INPROC_SERVER, IID_IOhioManager,
reinterpret_cast<void**>
(&pIOhioManager));

if (FAILED(hr)) {…} //error

BSTR bstrHEProfile, bstrSessionName;
bstrSessionName = SysAllocString
(OLESTR("aix"));
bstrHEProfile = SysAllocString
(OLESTR("C:\\Aix.hep"));

pIOhioManager->OpenSession
(l_bstrHEProfile, bstrSessionName,
&pIOhioSession);

SysFreeString(bstrHEProfile);
SysFreeString(bstrSessionName);
```

```
pIOhioSession->Connect();

pIOhioSession->Disconnect();
```

Method: IOhioSession::isConnected

This method indicates whether the OhioSession object is connected to a host.

Basic Syntax Boolean = OhioSession.**isConnected**

C++ Syntax HRESULT IOhioSession::**isConnected**([out, retval] VARIANT_BOOL * *outValue*);

Parameters *outValue*—A value of VARIANT_TRUE indicates that the object is connected to a host. A value of VARIANT_FALSE indicates that the object is not connected to a host.

Basic Example

```
Dim OSession As OhioSession

Dim bIsConnected As Boolean

. . .

bIsConnected = OSession.isConnected

If (bIsConnected = True) Then

‘connected

Else

‘not connected

End If
```

C++ Example

```
IOhioSession* pIOhioSession;

VARIANT_BOOL bConnected;

. . .

pIOhioSession->isConnected(&bConnected);

if (bConnected == VARIANT_FALSE)

{

printf("Session disconnected.\n");

}
```

Method: IOhioSession::OnSessionChanged

This method is called when the state of the session changes.

Basic Syntax OhioSession.**OnSessionChanged**(*inState* As OHIO_STATE)

C++ Syntax HRESULT IOhioSession::**OnSessionChanged**([in] OHIO_STATE *inState*);

Parameters *inState*—The changed state of the session.

Basic Example

```
Dim WithEvents oIOhioScr As
HEOHIOLib.OhioScreen
Dim WithEvents oIOhioSess As
HEOHIOLib.OhioSession
Dim OManager As OhioManager
……
Private Sub Form_Load()
strProfile = “C:\\Aix.hep”
strSessionName = “aix”
Set OManager = CreateObject
("HEOhio.OhioManager")
Set OSession = OManager.OpenSession
(strProfile, strSessionName)
Set oIOhioSess = Osession
End Sub
……
Private Sub oIOhioSess_OnSessionChanged
(ByVal inState As HEOHIOLib.OHIO_STATE)
Debug.Print “OnSessionChanged called.”
If (inState = OHIO_STATE_CONNECTED) Then
Debug.Print “Connected”
Else
Debug.Print “Disconnected”
End If
End Sub
```

Related Topics

OHIO_STATE Data Type

Property: IOhioSession::CanChangeScreen

This property indicates whether you or the host can change the contents of the screen.

Basic Syntax

Boolean = OhioSession.**CanChangeScreen**

OhioSession.**CanChangeScreen** = Boolean

C++ Syntax

HRESULT IOhioSession::**get_CanChangeScreen**([out, retval] VARIANT_BOOL * *pVal*);

HRESULT IOhioSession::**put_CanChangeScreen**([in] VARIANT_BOOL *newVal*);

Parameters

pVal—The system returns a value of VARIANT_TRUE if you or the host can change the screen contents. It returns VARIANT_FALSE if you or the host cannot change the screen contents.

newVal—A value of VARIANT_TRUE indicates that you or the host can change the screen contents. A value of VARIANT_FALSE indicates that you or the host cannot change the screen contents.

Basic Example

```
Dim bOSession As OhioSession

Dim bCanChangeScreen As Boolean

. . .

bCanChangeScreen=OSession.CanChangeScreen

If (bCanChangeScreen = True) Then

. . .

Else

. . .

End If

‘Set the value of OSession.CanChangeScreen

‘to False as default

OSession.CanChangeScreen = False
```

C++ Example

```
IOhioSession* pIOhiosession;

VARIANT_BOOL bCanChangeSceen;

. . .

HRESULT hr = pIOhioSession->

get_CanChangeScreen(&bCanChangeSceen);

if (FAILED(hr)) {…} //error

if (bCanChangeSceen)

printf("Can change screen.\n");

else

printf("Cannot change screen.\n");

//set to false

bCanChangeSceen = FALSE;

pIOhioSession->put_CanChangeScreen

(bCanChangeScreen);
```

Property: IOhioSession::ConfigurationResource

This property returns a vendor-specific string (Hummingbird profile) used to indicate configuration information.

Basic Syntax
String = OhioSession.**ConfigurationResource**

C++ Syntax
HRESULT IOhioSession::**get_ConfigurationResource**([out, retval] BSTR * *pVal*);

Parameters
pVal—The returned vendor-specific string.

Basic Example

```
Dim OSession As OhioSession

Dim strConfigRes As String

strConfigRes=OSession.ConfigurationResource()
```

C++ Example

```
IOhioSession* pIOhioSession;

. . .

BSTR bstrConfigurationResource;

HRESULT hr = pIOhioSession->

get_ConfigurationResource

(&bstrConfigurationResource);

if (FAILED(hr)) {…} //error

SysFreeString(bstrSessName);
```

Property: IOhioSession::Screen

This property returns the OhioScreen object for the session.

Basic Syntax
OhioScreen = OhioSession.**Screen**

C++ Syntax
HRESULT IOhioSession::**get_Screen**([out, retval] IOhioScreen * * *pVal*);

Parameters
pVal—The returned address of the IOhioScreen object.

Basic Example

```
Dim OSession As OhioSession

Dim OScreen As OhioScreen

. . .

Set OScreen = OSession.Screen
```

C++ Example

```
IOhioScreen* pIOhioScreen=NULL;

IOhioPosition* pIOhioPosition=NULL;

. . .

BSTR bstrWaitFor;

bstrWaitFor = SysAllocString

(OLESTR("login"));

HRESULT hr = pIOhioSession->get_Screen

(&pIOhioScreen);

if (SUCCEEDED(hr))

{

hr = pIohioScreen (pIohioScreen->
```

```
WaitForString(bstrWaitFor,24,1,2,20,

TRUE, &pIOhioPosition);

if (FAILED(hr)) {…} //error

}

SysFreeString(bstrWaitFor);
```

Property: IOhioSession::SessionName

This property returns a unique name associated with an OhioSession object.

Basic Syntax
String = OhioSession.**SessionName**

C++ Syntax
HRESULT IOhioSession::**get_SessionName**([out, retval] BSTR * *pVal*);

Parameters
pVal—The returned name of the IOhioSession object.

Basic Example

```
Dim OSess As OhioSession

Dim strSessName As String

. . .

strSessName = OSess.SessionName
```

C++ Example

```
IOhoSession* pIOhioSession;

. . .

BSTR bstrSessName;

HRESULT hr = pIOhioSession->

get_SessionName(&bstrSessName);

if (FAILED(hr)) {…} //error

USES_CONVERSION;

char* szSessName = OLE2A(bstrSessName);

printf("Session Name is %s\n",szSessName);

SysFreeString(bstrWaitFor);
```

Property: IOhioSession::SessionType

This property returns the session type for the OhioSession object.

Basic Syntax
OHIO_TYPE = OhioSession.**SessionType**

C++ Syntax
HRESULT IOhioSession::**get_SessionType**([out, retval] OHIO_TYPE * *pVal*);

Parameters
pVal—The returned session type for the IOhioSession object.

Basic Example

```
Dim OSession As OhioSession

Dim OType As OHIO_TYPE

. . .

OType = OSession.SessionType

If (OType = OHIO_TYPE_VT) Then

. . .

Else

. . .

End If
```

C++ Example

```
IOhioSession* pIohioSession;

OHIO_TYPE pSessionType;

. . .

HRESULT hr = pIOhioSession->

get_SessionType(&pSessionType);

if (FAILED(hr)) {…} //error

if (pSessionType==OHIO_TYPE_VT)

. . .

else

. . .
```

Related Topics

OHIO_TYPE Data Type

OhioScreen Interface

The OhioScreen interface is the host's virtual screen. It contains all the characters and attributes that would be seen on a traditional emulator screen. This interface is the primary object for text-based interactions with the host.

The interface provides methods such as manipulating text, searching the screen, sending keystrokes to the host, and handling the cursor. It lets you request an object that contains a collection of fields. Specifically, it can return the OIA (Operator Information Area) object.

Note: You can obtain an OhioScreen object from the Screen property of an instance of OhioSession.

The data on the screen is maintained in a series of planes, which can be accessed by various methods within the OhioScreen interface. Most of the methods in this interface work with the text plane, which contains the actual characters in the presentation place. The remaining planes (color, field, and extended) contain the corresponding attributes for each character in the text plane.

Methods

The OhioScreen interface consists of the following methods:

FindString

GetData

OnCursorMoved

OnScreenChanged

OnSizeChanged

PutString

SendKeys

WaitForInput

WaitForString

WaitIdle

Properties

The OhioScreen interface consists of the following properties:

Columns

Cursor

Fields

OIA

Rows

String

Method: IOhioScreen::FindString

This method searches the text plane for the target string. If it finds the string, the method returns an OhioPosition object containing the target location. If it does not find the string, the method returns a null. The target string must be completely contained in the target area for the search to be successful.

Basic Syntax

OhioScreen.**FindString**(*inText* As String, *inStart* As IOhioPosition, *inLen* As Long, *inDir* As OHIO_DIRECTION, *inIgnoreCase* As Boolean) As OhioPosition

C++ Syntax

HRESULT IOhioScreen::**FindString**([in] BSTR *inText*, [in] IOhioPosition * *inStart*, [in] long *inLen*, [in] OHIO_DIRECTION *inDir*, [in] VARIANT_BOOL *inIgnoreCase*, [out, retval] IOhioPosition * * *outPos*);

Parameters

inText—The target string.

inStart—The row and column where the search is to start. The position is inclusive.

inLen—The length from the starting position to include in the search.

inDir— An OHIO_DIRECTION data type value.

inIgnoreCase—A value of VARIANT_TRUE indicates that the search ignores case sensitivity. A value of VARIANT_FALSE indicates that the search is case-sensitive.

outPos—The returned address of the IOhioPosition object.

Basic Example

```
Dim OSession As OhioSession
Dim OScreen As OhioScreen
Dim ODir As OHIO_DIRECTION
Dim OPos, OPosStart As OhioPosition
Dim iLen As Long
. . .
ODir = OHIO_DIRECTION_FORWARD
ILen = 100

Set OPosStart = OSession.CreateOhioPosition(0, 0)
Set OPos = OScreen.FindString("login", OPosStart, 100, ODir, True)

If (OPos Is Nothing) Then
‘String not found
Else
‘Found String
End If
```

C++ Example

```
IOhioSession* pIOhioSession;
IOhioScreen* pIOhioScreen=NULL;
IOhioPosition* pIOhioPosition=NULL;
. . .
BSTR bstrWaitFor;
bstrWaitFor = SysAllocString(OLESTR("login"));
pIOhioSession->get_Screen(&pIOhioScreen);
HRESULT hr = pIOhioSession->
CreateOhioPosition
(20,1,&pIOhioPosition);
if (FAILED(hr)) {…} //error
pIOhioScreen->FindString(bstrWaitFor
,pIOhioPosition, 1000,
OHIO_DIRECTION_FORWARD, TRUE,
&pIOhioPosition);

if (pIOhioPosition==NULL)
//String not found
```

```
else

…



SysFreeString(bstrWaitFor);
```

Related Topics

OHIO_DIRECTION Data Type

Method: IOhioScreen::GetData

This method returns a byte array from the text, color, field, or extended plane of the virtual screen by specifying the starting and ending position.

Basic Syntax

OhioScreen.**GetData**(*inStart* As IOhioPosition, *inEnd* As IOhioPosition, *inPlane* As OHIO_PLANE) As VARIANT

C++ Syntax

HRESULT IOhioScreen::**GetData**([in] IOhioPosition * *inStart*, [in] IOhioPosition * *inEnd*, [in] OHIO_PLANE *inPlane*, [out, retval] VARIANT * *outData*);

Parameters

inStart—The row and column where the search is to begin. The position is inclusive.

inEnd—The row and column where the search is to end. The position is inclusive.

inPlane—The type of plane of the virtual screen.

outData—The data contained in the plane.

Basic Example

```
Dim OSession As OhioSession

Dim OScreen As OhioScreen

Dim OPosEnd As OhioPosition

Dim strPlane As String

Dim ByteArray As Variant

. . .

Set OPosStart = OSession.CreateOhioPosition(0, 0)

Set OPosEnd = OSession.CreateOhioPosition(24, 1)

ByteArray = OScreen.GetData(OPosStart, OPosEnd, OHIO_PLANE_TEXT)
```

C++ Example

```
IOhioSession* pIOhioSession;

IOhioScreen* pIOhioScreen;

IOhioPosition* pOhioPosStart, pOhioPosEnd;

VARIANT ByteArray;

. . .

pIOhioSession>CreateOhioPosition(0,0,&pOhioPosStart);

pIOhioSession->CreateOhioPosition(24,1,

&pOhioPosEnd);

pIOhioScreen->GetData(pOhioPosStart,

pOhioPosEnd, OHIO_PLANE_TEXT,

&ByteArray);
```

Related Topics

OHIO_COLOR Data Type

OHIO_EXTENDED Data Type

OHIO_FIELD Data Type

OHIO_PLANE Data Type

Method: IOhioScreen::OnCursorMoved

This method is called when the cursor moves.

Basic Syntax OhioScreen.**OnCursorMoved**(*inNewPosition* As IOhioPosition)

C++ Syntax HRESULT IOhioScreen::**OnCursorMoved**([in] IOhioPosition * * *inNewPosition*);

Parameters *inNewPosition*—The address of the new cursor position.

Basic Example

```
Dim WithEvents oIOhioScr As
HEOHIOLib.OhioScreen
Dim OManager As OhioManager
Dim OSession As OhioSession
……
Private Sub Form_Load()
strProfile = “C:\\Aix.hep”
strSessionName = “Aix”
Set OManager = CreateObject
("HEOhio.OhioManager")
Set OSession = OManager.OpenSession
(strProfile, strSessionName)
Set oIOhioScr = OSession.Screen
End Sub
……
Private Sub oIOhioScr_OnCursorMoved
(ByVal inNewPosition As
HEOHIOLib.IOhioPosition)
Debug.Print(vbCrLf & “OnCursorMoved
called”)
Debug.Print(vbCrLf & " inUpdate: " &
inUpdate)
Debug.Print(vbCrLf & " newPos (rc): " &
inNewPosition.Row & ", " &
```

```
inNewPosition.Column)

End Sub
```

Method: IOhioScreen::OnScreenChanged

This method is called when the screen changes.

Basic Syntax

OhioScreen.**OnScreenChanged**(inUpdate As OHIO_UPDATE, *inStart* As IOhioPosition, *inEnd* As IOhioPosition)

C++ Syntax

HRESULT IOhioScreen::**OnScreenChanged**([in] OHIO_UPDATE *inUpdate*, [in] IOhioPosition * *inStart*, [in] IOhioPosition * *inEnd*);

Parameters

inUpdate—Whether you or the host changed the screen.

inStart—The starting position of the location where the screen changed.

inEnd—The ending position of the location where the screen changed.

Basic Example

```
Dim WithEvents oIOhioScr As

HEOHIOLib.OhioScreen

Dim OManager As OhioManager

Dim OSession As OhioSession

. . .

Private Sub Form_Load()

strProfile = “C:\\Aix.hep”

strSessionName = “aix”

Set OManager = CreateObject

("HEOhio.OhioManager")

Set OSession = OManager.OpenSession

(strProfile, strSessionName)

Set oIOhioScr = OSession.Screen

End Sub

. . .

Private Sub

oIOhioScr_OnScreenChanged(ByVal

inUpdate As HEOHIOLib.OHIO_UPDATE,

ByVal inStart As

HEOHIOLib.IOhioPosition, ByVal

inEnd As HEOHIOLib.IOhioPosition)

Debug.Pring (vbCrLf &

“OnScreenChanged called")

Debug.Pring (vbCrLf & vbCrLf & "

inUpdate: " & inUpdate)
```

```
Debug.Pring (vbCrLf & vbCrLf & "
instart (rc): " & inStart.Row & ", " &
inStart.Column)
Debug.Pring (vbCrLf & vbCrLf & "
inEnd (rc): " & inEnd.Row & ", " &
inEnd.Column)
End Sub
```

Related Topics

OHIO_UPDATE Data Type

Method: IOhioScreen::OnSizeChanged

This method is called when the size of the screen changes.

Basic Syntax
OhioScreen.**OnSizeChanged**(*inNewRows* As Long, *inNewColumns* As Long)

C++ Syntax
HRESULT IOhioScreen::**OnSizeChanged**([in] long *inNewRows*, [in] long *inNewColumns*);

Parameters
inNewRows—The number of rows in the changed screen.
inNewColumns—The number of columns in the changed screen.

Basic Example

```
Dim WithEvents oIOhioScr As
HEOHIOLib.OhioScreen
Dim OManager As OhioManager
Dim OSession As OhioSession
……
Private Sub Form_Load()
strProfile = “C:\\Aix.hep”
strSessionName = “aix”
Set OManager = CreateObject
("HEOhio.OhioManager")
Set OSession = OManager.OpenSession
(strProfile, strSessionName)
Set oIOhioScr = OSession.Screen
End Sub
……
Private Sub oIOhioScr_OnSizeChanged(ByVal
inNewRows As Long, ByVal inNewColumns As
Long)
Debug.Print (vbCrLf & " OnSizeChanged
called")
```

```
Debug.Pring (vbCrLf & vbCrLf & "
inNewRows: " & inUpdate)
Debug.Pring (vbCrLf & vbCrLf & "
inNewColumns: " & inUpdate)
End Sub
```

Method: IOhioScreen::PutString

This method sends a string to the virtual screen at the specified location. The string displays in only unprotected fields; any parts of the string in protected fields are discarded.

Basic Syntax
OhioScreen.**PutString**(*inText* As String, *inStart* As IOhioPosition)

C++ Syntax
HRESULT IOhioScreen::**PutString**([in] BSTR *inText*, [in] IOhioPosition * *inStart*);

Parameters
inText—The string to place in the virtual screen.
inStart—The starting position of the string.

Basic Example

```
Dim OScreen As OhioScreen
Dim OPos As OhioPosition
. . .
‘get current cursor location
Set OPos = OScreen.Cursor
‘send to current cursor location
OScreen.PutString "jack123", OPos
‘OR
‘Easier way to send to current cursor
‘location
OScreen.PutString "jack123", Nothing
```

C++ Example

```
//Send string to specific cursor location
IOhioSession* pIOhioSession;
IOhioScreen* pIOhioScreen;
IOhioPosition* pIOhioPos;
. . .
HRESULT hr = pIOhioSession->
CreateOhioPosition(0, 0, &pOhioPos);
if (FAILED(hr)) {…} //error

BSTR bstrText;
bstrText=SysAllocString(OLESTR("jack123"));
pIOhioScreen->PutString(bstrText,
```

```
pOhioPos);


//Send string to current cursor location
pIOhioScreen->PutString(bstrText,NULL);


SysFreeString(bstrText);
```

Method: IOhioScreen::SendKeys

This method sends a string of keys to the virtual screen as if keystrokes were being typed from the keyboard. The keystrokes are sent to the location you specify. If you do not provide a location, the keystrokes are sent to the current cursor location.

Basic Syntax OhioScreen.**SendKeys**(*inText* As String, *inPos* As IOhioPosition)

C++ Syntax HRESULT IOhioScreen::**SendKeys**([in] BSTR *inText*, [in] IOhioPosition * *inPos*);

Parameters *inText*—The text that is typed from the keyboard.
inPos—The position where the keystrokes will appear.

Basic Example

```
Dim OScreen As OhioScreen

Dim OPos As OhioPosition

. . .

‘get current cursor location

Set OPos = OScreen.Cursor

OScreen.SendKeys "ls -lrt[enter]", OPos

if (FAILED(hr)) {…} //error

‘Tip -When Nothing is the second

‘parameter, by default the keys are sent

‘to the current cursor location

OScreen.SendKeys "ls -lrt[enter]", Nothing

‘send the [pf1] key

Oscreen.SendKeys [pf1], Nothing
```

C++ Example

```
IOhioScreen* pIOhioScreen
IOhioSession* pIOhioSession
IOhioPosition* pOhioPos;
. . .
//if the second parameter is NULL, the
//keys will be sent to the current cursor
//location
BSTR bstrText, bstrEnter;
bstrText=SysAllocString(OLESTR("tester"));
bstrEnter=SysAllocString(OLESTR("[enter]"));
HRESULT hr = pIOhioScreen->SendKeys
(bstrText,NULL);
if (FAILED(hr)) {…} //error
hr = pIOhioScreen->SendKeys
(bstrEnter,NULL);
if (FAILED(hr)) {…} //error
//send both strings together
bstrText= SysAllocString(OLESTR("tester[enter]"));
pIOhioScreen->SendKeys(bstrText,NULL);

//send to a cursor location different from
//the current location
pIOhioSession->CreateOhioPosition(0, 0, &pOhioPos);
hr = pIOhioScreen->SendKeys
(bstrText,pOhioPos);
if (FAILED(hr)) {…} //error

SysFreeString(bstrText);
SysFreeString(bstrEnter);
```

Related Topics

Mnemonic Keywords for Method IOhioScreen.SendKeys

Mnemonic Keywords

In the method OhioScreen.SendKeys, HostExplorer sends a string of keys so that it appears on the virtual screen. You can use mnemonic keywords to send special

functions or AID keys.

The following rules apply to mnemonic keywords:

· Enclose keywords in square brackets ([ ]).

· Keywords are not case-sensitive. For example, [Attn] is identical to [attn].

· Square brackets that should appear as literal values need to be escaped by doubling them (for example,]]). The first bracket escapes the second.

Examples of strings for the SendKeys method are as follows:

abc [pf1] Sends the characters a-b-c

[backspace] [[x]] Sends the 3270 backspace key followed by three “[x]” characters.

xyz[[[CLEAR] Sends the characters x-y-z-[followed by the 3270 Clear AID key.

The following list contains mnemonic keywords that are valid for the method OhioScreen.SendKeys:

Mnemonic 3270 Function Description

[attn] Attention AID key

[clear] Clear AID key

[cursorselect] Cursor Select AID key

[enter] Enter AID key

[sysreq] System Request

[pa1] Program Attention 1 AID key

[pa2] Program Attention 2 AID key

[pa3] Program Attention 3 AID key

[pf1] Program Function 1 AID key

[pf2] Program Function 2 AID key

[pf3] Program Function 3 AID key

[pf4] Program Function 4 AID key

[pf5] Program Function 5 AID key

[pf6] Program Function 6 AID key

[pf7] Program Function 7 AID key

[pf8] Program Function 8 AID key

[pf9] Program Function 9 AID key

[pf10] Program Function 10 AID key

[pf11] Program Function 11 AID key

[pf12] Program Function 12 AID key

[pf13] Program Function 13 AID key

[pf14] Program Function 14 AID key

[pf15] Program Function 15 AID key

[pf16] Program Function 16 AID key

[pf17] Program Function 17 AID key

[pf18] Program Function 18 AID key

[pf19] Program Function 19 AID key

[pf20] Program Function 20 AID key

[pf21] Program Function 21 AID key

[pf22] Program Function 22 AID key

[pf23] Program Function 23 AID key

[pf24] Program Function 24 AID key

[tab] Tab forward to next unprotected field

[backtab] Tab backward to previous unprotected field

[up] Cursor up

[down] Cursor down

[right] Cursor right

[left] Cursor left

[fastup] Cursor up two rows

[fastdown] Cursor down two rows

[fastright] Cursor right two positions

[fastleft] Cursor left two positions

[home] Home

[newline] New line—move to first unprotected field on next or

subsequent line

[reset] Reset—clear keyboard lock and clear insert mode

[insert] Insert mode—turns on insert mode for all subsequent

keystrokes until [reset]

[backspace] Destructive Backspace—move cursor one position left, delete character, shift remainder of field one position left

[delete] Delete at cursor, shift remainder of field one position left

[eraseinput] Erase input—clear all unprotected fields

[eraseof] Erase from cursor to end of field, inclusive

[dup] Duplicate

[fieldmark] Field Mark

Method: IOhioScreen::WaitForInput

This method allows HostExplorer to wait for the keyboard to be unlocked.

Basic Syntax
OhioScreen.**WaitForInput**(*inTimeOut* As Long)

C++ Syntax
HRESULT IOhioScreen::**WaitForInput**(long *inTimeOut*, IOhioPosition * * *outPos*);

Parameters
inTimeOut—The specified expiry time (in seconds).

outPos—The returned value, indicating the current cursor position of the session. If the time expires, the value returned is null.

Basic Example

```
Dim OScreen As OhioScreen

Dim OPos As OhioPosition

Dim iTimeOut As Long

. . .

iTimeOut = 10

Set OPos = OScreen.WaitForInput(iTimeOut)



If (OPos Is Nothing) Then

Debug.Print “null value returned”

End If
```

Method: IOhioScreen::WaitForString

This method allows HostExplorer to wait for the specified string to appear on the screen.

Basic Syntax
OhioScreen.**WaitForString**(*inString* As String, *inRow* As Integer, *inCol* As Integer, *inFlag* As Long, *inTimeOut* As Long, inIgnoreCase As Boolean) As OhioPosition

C++ Syntax
HRESULT IOhioScreen::**WaitForString**(BSTR *inString*, short *inRow*, short *inCol*, long *inFlag*, long *inTimeOut*, VARIANT_BOOL *inIgnoreCase*, IOhioPosition * * *outPos*);

Parameters
inString—The specified string.

inRow—The row that contains the specified string.

inCol—The column that contains the specified string.

inFlag—If you set this parameter to 1, HostExplorer searches for the string in the row specified in the inRow parameter only. If you set this parameter to 2, HostExplorer searches for the string starting from the row specified in the inRow parameter.

inTimeOut—The specified expiry time (in seconds).

inIgnoreCase—A value of VARIANT_TRUE indicates that the search is not case-sensitive. A value of VARIANT_FALSE indicates that the search is case-sensitive.

outPos—The returned value, indicating the position of the string. If the time has expired, or if no string is found, the value returned is null.

Basic Example

```
Dim OPos As OhioPosition
Dim iRow, iCol, iFlag As Integer
Dim iTimeOut As Long
Dim bIgnoreCase As Boolean
Dim searchStr As Str
. . .
iRow = 24
iCol = 1
iFlag = 2
iTimeOut = 100
bIgnoreCase = True
searchStr = “Password”
Set OPos = OScreen.WaitForString
(searchStr, iRow, iCol, iFlag, iTimeOut,
bIgnoreCase)

If (OPos Is Nothing) Then
‘String not found
Else
‘String found
End If
```

C++ Example

```
IOhioSession* pIOhioSession;
IOhioScreen* pIOhoScreen;
IOhioPosition* pIOhioPosition=NULL;
. . .
BSTR bstrWaitFor;
bstrWaitFor = SysAllocString
(OLESTR("login"));
HRESULT hr = pIOhioSession->get_Screen
(&pIOhioScreen);
if (FAILED(hr)) {…} //error
pIOhioScreen>WaitForString(bstrWaitFor,24,
1,2,20,TRUE,&pIOhioPosition);

if (pIOhioPosition==NULL)
```

```
//String not found

else

//String found

SysFreeString(bstrWaitFor);
```

Method: IOhioScreen::WaitIdle

This method allows the session to idle for a specified number of seconds.

Basic Syntax OhioScreen.**WaitIdle**(*inIdleTime* As Long)

C++ Syntax HRESULT IOhioScreen::**WaitIdle**(long *inIdleTime*);

Parameters *inIdleTime*—The number of seconds that the session is idle.

Basic Example

```
Dim iTimeOut As Long

Dim OScreen As OhioScreen

. . .

iTimeOut = 10

OScreen.WaitIdle(iTimeOut)
```

Property: IOhioScreen::Columns

This property returns the number of columns in the virtual screen.

Basic Syntax Long = OhioScreen.**Columns**

C++ Syntax HRESULT IOhioScreen::**get_Columns**([out, retval] long * *pVal*);

Parameters *pVal*—The returned number of columns in the screen.

Basic Example

```
Dim OScreen As OhioScreen

Dim nCol As Long

. . .

nCol = OScreen.Columns

strText ="number of columns is " & nCol
```

C++ Example

```
IOhioScreen* pIOhioScreen

. . .

long numCols;

HRESULT hr = pIOhioScreen->get_Columns

(&numCols);

if SUCCEEDED(hr))

printf("number of cols is %d",numCols);
```

Property: IOhioScreen::Cursor

This property returns or sets the location of the cursor in the virtual screen. The OhioPosition object contains the row and column of the cursor.

Basic Syntax

OhioPosition = OhioScreen.**Cursor**

OhioScreen.**Cursor** = IOhioPosition

C++ Syntax

HRESULT IOhioScreen::**get_Cursor**([out, retval] IOhioPosition * * *pVal*);

HRESULT IOhioScreen::**put_Cursor**([in] IOhioPosition * *newVal*);

Parameters

pVal—The returned address of the cursor position.

newVal—The address of the cursor position that you set.

Basic Example

```
‘Get the current cursor position

Dim OScreen As OhioScreen

Dim OSession As OhioSession

Dim OPos As OhioPosition

. . .

Set OPos = OScreen.Cursor

‘Set the current cursor position

Set OPos =

OSession.CreateOhioPosition(0, 0)

OScreen.Cursor = OPos
```

C++ Example

```
//get current cursor position

IOhioSession* pIOhioSession;

IOhioScreen* pIOhioScreen;

IOhioPosition* pOhioPos;

. . .

pIOhioScreen->get_Cursor(&pOhioPos);

//set current cursor position

pIOhioSession->CreateOhioPosition(0, 0,

&pOhioPos);

pIOhioScreen->put_Cursor(pOhioPos);
```

Property: IOhioScreen::Fields

This property returns the OhioFields object associated with the virtual screen, providing another way to access the data on the screen. The OhioFields object contains a static view of all the fields in the current virtual screen. Fields provide methods for interpreting the data in the non-text planes.

Note: Zero-length fields (caused by adjacent field attributes) are not returned in the OhioFields collection. For unformatted screens, the returned collection contains only one OhioField object, which contains the whole virtual screen.

Basic Syntax

OhioFields = OhioScreen.**Fields**

C++ Syntax

HRESULT IOhioScreen::**get_Fields**([out, retval] IOhioFields * * *pVal*);

Parameters

pVal—The returned address of the IOhioFields object.

Basic Example

```
Dim OScreen As OhioScreen

Dim OFields As OhioFields

Dim OField As OhioField

Dim iFieldCount As Integer

. . .

Set OFields = OScreen.Fields

iFCount = 0

For Each OField In OFields

iFieldCount = iFieldCount + 1

Next

strText = strText & "Number of Fields is " & str(iFieldCount)
```

C++ Example

```
long lIndex = 1;

long lLen;

IOhioScreen* pIOhioScreen;

IOhioFields* pIOhioFields;

IOhioField* pIOhioField;

. . .

HRESULT hr = pIOhioScreen->get_Fields

(&pIOhioFields);

if (FAILED(hr)) {…} //error

pIOhioFields->get_Item

(lIndex,&pIOhioField);

if (pIohioField ! = NULL)

{

pIOhioField->get_Length(&lLen);

printf("length of field is %d\n",lLen);

}

else

//error
```

Property: IOhioScreen::OIA

This property returns the OhioOIA object associated with the virtual screen. You can use this object to query the status of the Operator Information Area (OIA).

Basic Syntax OhioOIA = OhioScreen.**OIA**

C++ Syntax HRESULT IOhioScreen::**get_OIA**([out, retval] IOhioOIA * * *pVal*);

Parameters *pVal*—The returned address of the IOhioOIA object.

Basic Example

```
Dim OIA As OhioOIA

Dim bAlphaNumeric As Boolean

. . .

Set OIA = OScreen.OIA

bAlphaNumeric = OIA.AlphaNumeric

If (bAlphaNumeric = True) Then

‘cursor is in an alphanumeric field

Else

‘cursor is in a non-alphanumeric field

End If
```

C++ Example

```
IOhioOIA* pIOhioOIA;

VARIANT_BOOL bAlphanumeric;

. . .

HRESULT hr = pIOhioScreen->get_OIA

(&pIOhioOIA);

if (FAILED(hr)) {…} //error

pIOhioOIA->

get_Alphanumeric(&bAlphanumeric);

if (bAlphanumeric)

//cursor is in an alphanumeric field

else

//cursor is in a non-alphanumeric field
```

Property: IOhioScreen::Rows

This property returns the number of rows in the virtual screen.

Basic Syntax Long = OhioScreen.**Rows**

C++ Syntax HRESULT IOhioScreen::**get_Rows**([out, retval] long * *pVal*);

Parameters *pVal*—The returned number of rows in the screen.

Basic Example

```
Dim OScreen As OhioScreen

Dim nRows As Long

. . .

nRows = OScreen.Rows

strText = strText & "Rows =" & nRows
```

C++ Example

```
long lRows;

IOhioScreen* pIOhioScreen;

. . .

HRESULT hr = pIOhioScreen->get_Rows

(&lRows);

if (FAILED(hr)) {…} //error

printf("number of rows: %d\n",lRows);
```

Property: IOhioScreen::String

This property returns the entire text plane of the virtual screen as a string. This property returns all null characters and field attribute characters as blank space characters.

Basic Syntax String = OhioScreen.**String**

C++ Syntax HRESULT IOhioScreen::**get_String**([out, retval] BSTR * *pVal*);

Parameters *pVal*—The returned text plane.

Basic Example

```
Dim OScreen As OhioScreen

Dim strDisplayScreen As String

. . .

StrDisplayScreen = OScreen.String
```

C++ Example

```
IOhioScreen* pIOhioScreen;

BSTR bstrString;

. . .

HRESULT hr = pIOhioScreen->get_String

(&bstrString);

if (FAILED(hr)) {…} //error

USES_CONVERSION

printf("text plane: %s\n",OLE2A

(bstrString));

SysFreeString(bstrString);
```

OhioOIA Interface

The OhioOIA interface returns the Operator Information Area (OIA) object of a host session. The OhioOIA object contains the information displayed at the bottom of the screen, which provides the user with the session name, IP address, column and row numbers, and cursor position.

Properties

The OhioOIA interface consists of the following properties:

Alphanumeric

APL

CommCheckCode

Inputinhibited

InsertMode

MachineCheckCode

Numeric

Owner

ProgCheckCode

Property: IOhioOIA::Alphanumeric

This property indicates whether the cursor is in an alphanumeric field.

Basic Syntax Boolean = OhioOIA.**Alphanumeric**

C++ Syntax HRESULT IOhioOIA::**get_Alphanumeric**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the field that contains the cursor is an alphanumeric field. A returned value of VARIANT_FALSE indicates that the field that contains the cursor is not an alphanumeric field.

Basic Example

```
Dim OScreen As OhioScreen

Dim OIA As OhioOIA

Dim bAlphaNumeric As Boolean

. . .

Set OIA = OScreen.OIA

bAlphaNumeric = OIA.AlphaNumeric

If (bAlphaNumeric = True) Then

‘cursor is in an alphanumeric field

Else

‘cursor is in a non-alphanumeric field

End If
```

Property: IOhioOIA::APL

This property indicates whether the session is in A Program Language (APL) input mode.

Basic Syntax Boolean = OhioOIA.**APL**

C++ Syntax HRESULT IOhioOIA::**get_APL**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the session is in APL input mode. A returned value of VARIANT_FALSE indicates that the session is not in APL input mode.

Basic Example

```
Dim OScreen As OhioScreen

Dim OIA As OhioOIA

. . .

Set OIA = OScreen.OIA

Dim bAPL As Boolean

bAPL = OIA.APL

If (bAPL = True) Then

‘in APL input mode

Else

‘not in APL input mode

End If
```

C++ Example

```
IOhioScreen* pIOhioScreen;

IOhioOIA* pIOhioOIA;

VARIANT_BOOL bAPL;

. . .

pIOhioScreen->get_OIA(&pIOhioOIA);

pIOhioOIA->get_APL(&bAPL);

if (bAPL)

//in APL input mode

else

//not in APL input mode
```

Property: IOhioOIA::CommCheckCode

This property returns the communication check code if the InputInhibited property returns OHIO_INPUTINHIBITED_COMMCHECK.

Basic Syntax Long = OhioOIA.**CommCheckCode**

C++ Syntax HRESULT IOhioOIA::**get_CommCheckCode**([out, retval] long * *pVal*);

Parameters *pVal*—The returned communication check code.

Basic Example

```
Dim OIA As OhioOIA

Dim InputInhibited As OHIO_INPUTINHIBITED

Dim ChkCode As Long

. . .

InputInhibited = OIA.InputInhibited

If (InputInhibited =

OHIO_INPUTINHIBITED_COMMCHECK) Then

ChkCode = OIA.CommCheckCode

. . .

End If
```

C++ Example

```
IOhioOIA* pIOhioOIA;

long lCommCheckCode;

OHIO_INPUTINHIBITED InputInhibited;

. . .

pIOhioOIA->get_InputInhibited

(&InputInhibited);

if(InputInhibited == OHIO_INPUTINHIBITED_COMMCHECK)

{

pIOhioOIA->get_CommCheckCode

(&lCommCheckCode);

. . .

}
```

Related Topics

OHIO_INPUTINHIBITED Data Type

Property: IOhioOIA::InputInhibited

This property indicates whether input is inhibited. If input is inhibited, the system prohibits SendKeys or SendAid methods to the OhioScreen interface. The returned value can indicate the reason that input is inhibited. If input is inhibited for more than one reason, the highest value is returned.

Basic Syntax

```
OHIO_INPUTINHIBITED = OhioOIA.InputInhibited
```

C++ Syntax

```
HRESULT IOhioOIA::get_InputInhibited([out, retval] OHIO_INPUTINHIBITED *
pVal);
```

Parameters

pVal— The returned value, indicating whether input is inhibited.

Basic Example

```
Dim OIA As OhioOIA

Dim ChkCode As Long

Dim InputInhibited As OHIO_INPUTINHIBITED

. . .

InputInhibited = OIA.InputInhibited

If (InputInhibited = OHIO_INPUTINHIBITED_COMMCHECK) Then

‘input inhibited

ChkCode = OIA.CommCheckCode

…

End If
```

C++ Example

```
IOhioOIA* pIOhioOIA;

long lChkCode;

OHIO_INPUTINHIBITED InputInhib;

. . .

pIOhioOIA->get_InputInhibited

(&InputInhib);

if (InputInhib ==

OHIO_INPUTINHIBITED_COMMCHECK)

{

pIOhioOIA->get_CommCheckCode

(&lChkCode);

. . .

}
```

Related Topics

OHIO_INPUTINHIBITED Data Type

Property: IOhioOIA::InsertMode

This property indicates if the session is in Insert mode.

Basic Syntax Boolean = OhioOIA.**InsertMode**

C++ Syntax HRESULT IOhioOIA::**get_InsertMode**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the session is in Insert mode. A returned value of VARIANT_FALSE indicates that the session is not in Insert mode.

Basic Example

```
Dim OIA As OhioOIA

Dim bInsertMode As Boolean

. . .

bInsertMode = OIA.InsertMode

If (bInsertMode = True) Then

‘session in insert mode

Else

‘session not in insert mode

End If
```

C++ Example

```
IOhioScreen* pIOhioScreen;

IOhioOIA* pIOhioOIA;

VARIANT_BOOL bInsert;

. . .

pIOhioScreen->get_OIA(&pIOhioOIA);

pIOhioOIA->get_InsertMode(&bInsert);

if (bInsert)

//session in insert mode

else

//session not in insert mode
```

Property: IOhioOIA::MachineCheckCode

This property returns the machine check code if the InputInhibited property returns OHIO_INPUTINHIBITED_MACHINECHECK.

Basic Syntax Long = OhioOIA.**MachineCheckCode**

C++ Syntax HRESULT IOhioOIA::**get_MachineCheckCode**([out, retval] long * *pVal*);

Parameters *pVal*—The returned machine check code.

Basic Example

```
Dim OIA As OhioOIA

Dim InputInhibited As OHIO_INPUTINHIBITED

Dim ChkCode As Long

. . .

InputInhibited = OIA.InputInhibited

If (InputInhibited = OHIO_INPUTINHIBITED_MACHINECHECK) Then

ChkCode = OIA.MachineCheckCode

…

End If
```

C++ Example

```
IOhioOIA* pIOhioOIA;

long lChkCode;

OHIO_INPUTINHIBITED Inhib;

. . .

pIOhioOIA->get_InputInhibited(&Inhib);

if

(Inhib==OHIO_INPUTINHIBITED_MACHINECHECK)

{

pIOhioOIA->get_MachineCheckCode

(&lChkCode);

. . .

}
```

Related Topics

OHIO_INPUTINHIBITED Data Type

Property: IOhioOIA::Numeric

This property indicates whether the field that contains the cursor is a numeric-only field.

Basic Syntax

```
Boolean = OhioOIA.Numeric
```

C++ Syntax

```
HRESULT IOhioOIA::get_Numeric([out, retval] VARIANT_BOOL * pVal);
```

Parameters

pVal—A returned value of VARIANT_TRUE indicates that the field that contains the cursor is a numeric-only field. A returned value of VARIANT_FALSE indicates that the field is not a numeric-only field.

Basic Example

```
Dim OIA As OhioOIA

Dim bNumeric As Boolean

. . .

bNumeric = OIA.Numeric

If (bNumeric = True) Then

‘cursor is in a numeric-only field

End If
```

C++ Example

```
IOhioScreen* pIOhioScreen;

IOhioOIA* pIOhioOIA;

VARIANT_BOOL bNumeric;

. . .

HRESULT hr = pIOhioScreen->get_OIA

(&pIOhioOIA);

if (FAILED(hr)) {…} //error

pIOhioOIA->get_Numeric(&bNumeric);

if (bNumeric)

//cursor is in a numeric-only field

else

//cursor is not in a numeric-only field
```

Property: IOhioOIA::Owner

This property specifies the owner of the host connection.

Basic Syntax

```
OHIO_OWNER = OhioOIA.Owner
```

C++ Syntax

HRESULT IOhioOIA::**get_Owner**([out, retval] OHIO_OWNER * *pVal*);

Parameters

pVal—The returned owner of the host connection.

Basic Example

```
Dim OIA As OhioOIA

Dim Owner As OHIO_OWNER

. . .

Owner = OIA.Owner

If (Owner = OHIO_OWNER_UNKNOWN) Then

‘Add code

Else

‘Add code

End If
```

C++ Example

```
IOhioOIA* pIOhioOIA;

OHIO_OWNER OwnerOIA;

. . .

HRESULT hr = pIOhioOIA->get_Owner

(&OwnerOIA);

if (FAILED(hr)) {…} //error

if (OwnerOIA==OHIO_OWNER_UNKNOWN)

//Add code
```

Related Topics

OHIO_OWNER Data Type

Property: IOhioOIA::ProgCheckCode

This property returns the program check code if the InputInhibited property returns OHIO_INPUTINHIBITED_PROGCHECK.

Basic Syntax
Long = OhioOIA.**ProgCheckCode**

C++ Syntax
HRESULT IOhioOIA::**get_ProgCheckCode**([out, retval] long * *pVal*);

Parameters
pVal—The returned program check code.

Basic Example

```
Dim OIA As OhioOIA

Dim InputInhibited As OHIO_INPUTINHIBITED

Dim ChkCode As Long

. . .

InputInhibited = OIA.InputInhibited

If (InputInhibited = OHIO_INPUTINHIBITED_PROGCHECK) Then

ChkCode = OIA.ProgCheckCode

Else

‘handle other codes

End If
```

C++ Example

```
IOhioOIA* pIOhioOIA;

long lChkCode;

OHIO_INPUTINHIBITED InputInhibited;

. . .

HRESULT hr = pIOhioOIA->get_InputInhibited

(&InputInhibited);

if (FAILED(hr)) {…} //error

if (InputInhibited==

OHIO_INPUTINHIBITED_PROGCHECK)

pIOhioOIA->get_ProgCheckCode

(&lChkCode);
```

Related Topics

OHIO_INPUTINHIBITED Data Type

OhioFields Interface

The OhioFields interface contains a collection of the fields in the virtual screen. Each element of the collection is an instance of OhioField. Through this interface, you can iterate through the fields and find fields based on location and string.

You can access the OhioFields interface only through the OhioScreen interface using the Fields property.

Note: OhioFields is a static view of the virtual screen. It does not reflect your changes until you update the field list with a new view of the virtual screen using the Refresh method.

The OhioFields interface returns the number of fields in the screen.

Methods

The OhioFields interface consists of the following methods:

FindByPosition

FindByString

Refresh

Properties

The OhioFields interface consists of the following properties:

Count

Item

Method: IOhioFields::FindByPosition

This method searches the collection for the requested position and returns the OhioField object containing that position. If the position is not found, the method returns a null.

Basic Syntax	OhioFields.**FindByPosition**(*inPosition* As IOhioPosition) As OhioField
C++ Syntax	HRESULT IOhioFields::**FindByPosition**([in] IOhioPosition * *inPosition*, [out, retval] IOhioField * * *outField*);
Parameters	*inPosition*—The target row and column. *outField*—The returned address of the IOhioField object.
Basic Example	Dim OSession As OhioSession Dim OFields As OhioFields Dim OField As OhioField . . . Set OFields = OScreen.Fields Set OPos = OSession.CreateOhioPosition(15, 15) Set OField = OFields.FindByPosition(OPos)

C++ Example

```
long lLen;

IOhioSession* pIOhioSession;

IOhioPosition* inPos;

. . .

pIOhioSession->CreateOhioPosition

(24,1,&inPos);

HRESULT hr = pIOhioFields->FindByPosition

(inPos, &pIOhioField);

if (FAILED(hr)) {…} //error

pIOhioField->get_Length(&lLen);
```

Method: IOhioFields::FindByString

This method searches the collection for the requested string and returns the OhioField object containing that string. To be considered a match, the string must be contained completely within the field. If the string is not found, the method returns a null.

You can also specify case-sensitivity and whether the search is performed backward or forward.

Basic Syntax

OhioFields.**FindByString**(*inString* As String, *inStart* As IOhioPosition, *inLen* As Long, *inIgnoreCase* As Boolean) As OhioField

C++ Syntax

HRESULT IOhioFields::**FindByString**([in] BSTR *inString*, [in] IOhioPosition * *inStart*, [in] long *inLen*, [in]OHIO_DIRECTION *inDir*, [in] VARIANT_BOOL *inIgnoreCase*, [out, retval] IOhioField * * *outField*

Parameters

inString—The target string.

inStart—The row and column where the search is to start.

inLen—The length, from the starting position, to include in the search.

inDir—The OHIO_DIRECTION data type value.

inIgnoreCase—Whether the search is case-sensitive. VARIANT_TRUE indicates that case will be ignored. VARIANT_FALSE indicates that the search will be case-sensitive.

outField—The returned address of the OhioField object.

Basic Example

```
Dim OSession As OhioSession

Dim OFields As OhioFields

Dim OField As OhioField

. . .

Set OFields = OScreen.Fields

Set OPos = OSession.CreateOhioPosition(1, 1)

Set OField = OFields.FindByString("Next", OPos, 2000, OHIO_DIRECTION_FORWARD,
True)

If (OField Is Nothing) Then

‘the string is not found

Else

‘string found

End If
```

C++ Example

```
IOhioSession* pIOhioSession;

IOhioFields* pIOhioFields;

long lLen;

BSTR bstrSearch;

IOhioPosition* inPos;

. . .

bstrSearch=SysAllocString(OLESTR("test"));

pIOhioSession->CreateOhioPosition

(24,1,&inPos);

HRESULT hr = pIOhioFields->FindByString

(bstrSearch,inPos,400,

OHIO_DIRECTION_FORWARD,TRUE,&pIOhioField);

if (FAILED(hr)) {…} //error

pIOhioField->get_Length(&lLen);

SysFreeString(bstrSearch);
```

Method: IOhioFields::Refresh

This method updates the collection of OhioField objects, adding all OhioField objects in the current virtual screen to the collection. This method does not preserve indexing of OhioField objects.

Basic Syntax OhioFields.**Refresh**

C++ Syntax HRESULT IOhioFields::**Refresh**();

Parameters This method has no parameters.

Basic Example

```
Dim OSession As OhioSession

Dim OFields As OhioFields

Dim OField As OhioField

. . .

Set OPos = OSession.CreateOhioPosition(1, 1)

‘update Ofields collection before

‘accessing it

OFields.Refresh

Set OField = OFields.FindByPosition(OPos)
```

C++ Example

```
IOhioSession* pIOhioSession;
IOhioFields* pIOhioFields;
IOhioField* pIOhioField;
long lLen;
BSTR bstrSearch;
IOhioPosition* inPos;
. . .
bstrSearch=SysAllocString(OLESTR("test"));
pIOhioSession->CreateOhioPosition(24, 1,
&inPos);
pIOhioFields->Refresh();
HRESULT hr = pIOhioFields->FindByString
(bstrSearch,inPos,400,
OHIO_DIRECTION_FORWARD, TRUE,
&pIOhioField);
if (FAILED(hr)) {…} //error
pIOhioField->get_Length(&lLen);
SysFreeString(bstrSearch);
```

Property: IOhioFields::Count

This property returns the number of OhioField objects contained in the OhioFields collection.

Basic Syntax Long = OhioFields.**Count**

C++ Syntax HRESULT IOhioFields::**get_Count**([out, retval] long * *pVal*);

Parameters *pVal*—The returned number of IOhioField objects.

Basic Example

```
Dim OFields As OhioFields
Dim nCount As Integer
. . .
nCount = OFields.Count
strText = "Number of fields is " & nCount
```

C++ Example

```
IOhoScreen* pIOhioScreen;

long lcount;

IOhioFields* pIOhioFields;

. . .

HRESULT hr = pIOhioScreen->get_Fields

(&pIOhioFields);

if (FAILED(hr)) {…} //error

pIOhioFields->get_Count(&lcount);
```

Property: IOhioFields::Item

This property returns the OhioField object at the specified index (one-based indexing).

Basic Syntax

Long = OhioFields.**Item**
OhioFields.**Item** = Long

C++ Syntax

HRESULT IOhioFields::**get_Item**([in] long *inIndexOrKey*, [out, retval] IOhioField * * *pVal*);

Parameters

inIndexOrKey—The index of the returned IOhioField object.
pVal—The returned address of the IOhioField object.

Basic Example

```
Dim OField As OhioField

Dim strFieldText As String

. . .

Set OField = OFields.Item(4)

‘retrieve text from field number 4

strFieldText = OField.String
```

C++ Example

```
long lIndex = 1;

long lLen=0;

IOhioFields* pIOhioFields;

IOhioField* pIOhioField;

. . .

HRESULT hr = pIOhioScreen->get_Fields

(&pIOhioFields);

if (FAILED(hr)) {…} //error

pIOhioFields->get_Item

(lIndex,&pIOhioField);

pIOhioField->get_Length(&lLen);
```

OhioField Interface

The OhioField interface is a virtual-screen field that includes both data and attributes describing the field. The interface provides methods for accessing and manipulating field attributes and data.

Note: For VT terminals, the interface returns the entire screen because the VT terminal does not have fields.

You can access OhioField methods and properties only through the OhioFields interface.

Methods

The OhioField interface consists of the following method:

GetData

Properties

The OhioField interface consists of the following properties:

Attribute

End

HighIntensity

Length

Modified

Normal

Numeric

PenSelectable

Protected

Start

String

Method: IOhioField::GetData

This method lets you request different types of data, such as text, color, and field.

Basic Syntax	OhioField.**GetData**(*inPlane* As OHIO_PLANE) As Variant
C++ Syntax	HRESULT IOhioField::**GetData**([in] OHIO_PLANE *inPlane*, [out, retval] VARIANT * *outData*);
Parameters	*inPlane*—The type of plane. *outData*—The returned data contained in the plane.
Basic Example	Dim OField As OhioField Dim aByteArray As Variant . . . ByteArray = OField.GetData (OHIO_PLANE_TEXT)

C++ Example

```
IOhioField* pIOhioField;

VARIANT ByteArray;

OHIO_PLANE OhioPlane;

OhioPlane = OHIO_PLANE_TEXT;

pIOhioField->GetData(OhioPlane,

&ByteArray);
```

Related Topics

OHIO_PLANE Data Type

Property: IOhioField::Attribute

This property returns the attribute byte for the field.

Basic Syntax
OHIO_FIELD = OhioField.**Attribute**

C++ Syntax
HRESULT IOhioField::**get_Attribute**([out, retval] OHIO_FIELD * *pVal*);

Parameters
pVal—The returned attribute byte for the field.

Basic Example

```
Dim OField As OhioField

Dim OAtt As OHIO_FIELD

OAtt = OField.Attribute

Dim lResult As Long



‘check if an attribute is enabled with a

‘bitwise And

lResult = OAtt And

OHIO_FIELD_PEN_SELECTABLE

If (lResult = 0) Then

‘Attribute not enabled

Else

‘Attribute enabled

End If



‘Another way of doing the same thing

lResult = OAtt And

OHIO_FIELD_PEN_SELECTABLE

If (lResult = OHIO_FIELD_PEN_SELECTABLE) Then

‘Attribute enabled
```

```
Else
'Attribute not enabled
End If
```

C++ Example

```
IOhioField* pIOhioField;
long lResult;
OHIO_FIELD OAtt;
. . .
HRESULT hr = pIOhioField->
get_Attribute(&OAtt);
if (SUCCEEDED(hr)
{
lResult = OAtt &&
OHIO_FIELD_PEN_SELECTABLE;

if (lResult ==
OHIO_FIELD_PEN_SELECTABLE)
//Attribute enabled
else
//Attribute not enabled

//Another way of doing the same thing
lResult = OAtt &&
OHIO_FIELD_PEN_SELECTABLE;
if (lResult == 0)
//Attribute not enabled
else
//Attribute enabled
}
```

Related Topics

OHIO_FIELD Data Type

Property: IOhioField::End

This property returns the ending position of the field, which is the last character in the field. The position can range from 1 to the size of the virtual screen.

Basic Syntax OhioPosition = OhioField.**End**

C++ Syntax HRESULT IOhioField::**get_End**([out, retval] IOhioPosition * * *pVal*);

Parameters *pVal*—The returned value of the ending position of the field.

Basic Example

```
Dim OField As OhioField

Dim OPos As OhioPosition

Dim row, col As Long

. . .

Set OPos = OField.End

row = OPos.row

col = OPos.Column
```

C++ Example

```
IOhioField* pIOhioField;

IOhioPosition* PosEnd;

long row,col;

. . .

HRESULT hr = pIOhioField->get_End

(&PosEnd);

if (FAILED(hr)) {…} //error

PosEnd->get_Row(&row);

PosEnd->get_Column(&col);
```

Property: IOhioField::HighIntensity

This property indicates whether the field is high-intensity.

Basic Syntax Boolean = OhioField.**HighIntensity**

C++ Syntax HRESULT IOhioField::**get_HighIntensity**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the field is high-intensity. A returned value of VARIANT_FALSE indicates that the field is not high-intensity.

Basic Example

```
Dim OField As OhioField

Dim bHighIntensity As Boolean

. . .

bHighIntensity = OField.HighIntensity

If (bHighIntensity = True) Then

‘field is high-intensity

Else

‘field is not high-intensity

End If
```

C++ Example

```
IOhioField* pIOhioField;

VARIANT_BOOL bHighIntensity;

. . .

HRESULT hr=pIOhioField->get_HighIntensity

(&bHighIntensity);

if (FAILED(hr)) {…} //error

if (bHighIntensity)

//field is high-intensity

else

//field is not high-intensity
```

Property: IOhioField::Length

This property returns the length of the field. The length of a field can range from 1 to the size of the virtual screen.

Basic Syntax Long = OhioField.**Length**

C++ Syntax HRESULT IOhioField::**get_Length**([out, retval] long * *pVal*);

Parameters *pVal*—The returned value of the length of the field.

Basic Example

```
Dim OField As OhioField

Dim Length As Long

. . .

Length = OField.Length
```

C++ Example

```
IOhioField* pIOhioField;

Long lLen;

. . .

HRESULT hr = pIOhioField->get_Length

(&lLen);

if (FAILED(hr)) {…} //error
```

Property: IOhioField::Modified

This property indicates whether the field has been modified.

Basic Syntax Boolean = OhioField.**Modified**

C++ Syntax HRESULT IOhioField::**get_Modified**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the field has been modified. A returned value of VARIANT_FALSE indicates that the field has not been modified.

Basic Example

```
Dim OField As OhioField

Dim bModified As Boolean

. . .

bModified = OField.Modified

If (bModified = True) Then

‘field has been modified

Else

‘field not modified

End If
```

C++ Example

```
IOhioField* pIOhioField;

VARIANT_BOOL bModified;

. . .

HRESULT hr=pIOhioField->get_HighIntensity

(&bModified);

if (FAILED(hr)) {…} //error

if (bModified)

//field has been modified

else

//field not modified
```

Property: IOhioField::Normal

This property indicates whether the field is normal (that is, not protected and not high-intensity).

Basic Syntax Boolean = OhioField.**Normal**

C++ Syntax HRESULT IOhioField::**get_Normal**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the field is normal. A returned value of VARIANT_FALSE indicates that the field is not normal.

Basic Example

```
Dim OField As OhioField

Dim bNormal As Boolean

. . .

bNormal = OField.Normal

If (bNormal = True) Then

‘field is normal

Else

‘field is not normal

End If
```

C++ Example

```
IOhioField* pIOhioField;

VARIANT_BOOL bNormal;

. . .

HRESULT hr = pIOhioField->get_Normal

(&bNormal);

if (FAILED(hr)) {…} //error

if (bNormal)

//field is normal

else

//field is not normal
```

Property: IOhioField::Numeric

This property indicates whether the field is numeric-only.

Basic Syntax Boolean = OhioField.**Numeric**

C++ Syntax HRESULT IOhioField::**get_Numeric**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the field is numeric-only. A returned value of VARIANT_FALSE indicates that the field is not numeric-only.

Basic Example

```
Dim OField As OhioField

Dim bNumeric As Boolean

. . .

bNumeric = OField.Numeric

If (bNumeric = True) Then

‘field is numeric-only

Else

‘field is not numeric-only

End If
```

C++ Example

```
IOhioField* pIOhioField;

VARIANT_BOOL bNumeric;

. . .

HRESULT hr = pIOhioField->get_Numeric

(&bNumeric);

if (FAILED(hr)) {…} //error

if (bNumeric)

//field is numeric-only

else

//field is not numeric-onlly
```

Property: IOhioField::PenSelectable

This property indicates whether the field is pen-selectable.

Basic Syntax Boolean = OhioField.**PenSelectable**

C++ Syntax HRESULT IOhioField::**get_PenSelectable**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the field is pen-selectable. A returned value of VARIANT_FALSE indicates that the field is not pen-selectable.

Basic Example

```
Dim OField As OhioField

Dim bPenSelectable As Boolean

. . .

bPenSelectable = OField.PenSelectable

If (bPenSelectable = True) Then

‘field is pen-selectable

Else

‘field is not pen-selectable

End If
```

C++ Example

```
IOhioField* pIOhioField;

VARIANT_BOOL bPenSelectable;

. . .

HRESULT hr = pIOhioField->get_Numeric

(&bPenSelectable);

if (FAILED(hr)) {…} //error

if (bPenSelectable)

//field is pen-selectable

else

//field is not pen-selectable
```

Property: IOhioField::Protected

This property indicates whether the field is protected.

Basic Syntax Boolean = OhioField.**Protected**

C++ Syntax HRESULT IOhioField::**get_Protected**([out, retval] VARIANT_BOOL * *pVal*);

Parameters *pVal*—A returned value of VARIANT_TRUE indicates that the field is protected. A returned value of VARIANT_FALSE indicates that the field is not protected.

Basic Example

```
Dim OField As OhioField

Dim bProtected As Boolean

. . .

bProtected = OField.Protected

If (bProtected = True) Then

‘Field is protected

Else

‘Field is unprotected

End If
```

C++ Example

```
IOhioField* pIOhioField;

VARIANT_BOOL bProtected;

. . .

HRESULT hr = pIOhioField->get_Numeric

(&bProtected);

if (FAILED(hr)) {…} //error

if (bProtected)

//Field is protected

else

//Field is unprotected
```

Property: IOhioField::Start

This property returns the starting position of the field, which is the first character of the field. The position can range from 1 to the size of the virtual screen.

Basic Syntax OhioPosition = OhioField.**Start**

C++ Syntax HRESULT IOhioField::**get_Start**([out, retval] IOhioPosition * * *pVal*);

Parameters *pVal*—The returned starting position of the field.

Basic Example

```
Dim OField As OhioField

Dim OPos As OhioPosition

Dim row, col As Long

. . .

Set OPos = OField.Start

row = OPos.row

col = OPos.Column
```

C++ Example

```
IOhioField* pIOhioField;

IOhioPosition* PosStart;

long row,col;

. . .

HRESULT hr = pIOhioField->get_Start

(&PosStart);

if (FAILED(hr)) {…} //error

PosStart->get_Row(&row);

PosStart->get_Column(&col);
```

Property: IOhioField::String

This property returns or sets the text-plane data for the field as a string. If you set the property with a value shorter than the field, the system clears the rest of the field. If you set the property with a value longer than the field, the text is shortened. To view the changed text, you need to refresh the OhioFields collection.

Basic Syntax

String = OhioField.**String**
OhioField.**String** = String

C++ Syntax

HRESULT IOhioField::**get_String**([out, retval] BSTR * *pVal*);
HRESULT OhioField.**put_String**([in] BSTR *newVal*);

Parameters

pVal—The returned text-plane data for the field.
newVal—The data to appear in the text plane.

Basic Example

```
Dim OField As OhioField

Dim OFields As OhioFields

Dim strFieldText As String

. . .

Set OField = OFields(4)

strFieldText = OField.String
```

C++ Example

```
IOhioField* pIOhioField;

//get string

BSTR bstrString;

HRESULT hr = pIOhioField->get_String

(&bstrString);

if (FAILED(hr)) {…} //error

printf("field-get_string is: %s\n",OLE2A(bstrString));

SysFreeString(bstrString);



//put string

VARIANT_BOOL bProtected;

hr = pIOhioField->get_Protected
```

```
(&bProtected);
if (FAILED(hr)) {…} //error
if (bProtected)
//Add code
else
{
BSTR inBstr;
inBstr=SysAllocString(OLESTR
("sample text"));
hr = pIOhioField->put_String(inBstr);
if (FAILED(hr)) {…} //error
SysFreeString(inBstr);
}
```

OhioPosition Interface

The OhioPosition interface provides the row and column coordinates of the cursor position. You can create an OhioPosition by using the CreateOhioPosition method. 

The interface is used by the following sub-interfaces:

· OhioScreen

· OhioFields

· OhioField

Methods

The OhioPosition interface consists of the following method:

CreateOhioPosition

Properties

The OhioPosition interface consists of the following properties:

Column

Row

Method: IOhioPosition::CreateOhioPosition

This method allows you to create an OhioPosition object that can be passed to all interfaces. The object consists of a row and column coordinate.

Basic Syntax
OhioPosition.**CreateOhioPosition**(*inRow* As Long, *inCol* As Long) As OhioPosition

C++ Syntax
HRESULT IOhioPosition::**CreateOhioPosition** ([in] long *inRow*, [in] long *inCol*, [out, retval] IOhioPosition * * *outPosition*);

Parameters
inRow—The row coordinate.
inCol—The column coordinate.
outPosition—The address of the new OhioPosition object.

Basic Example

```
Dim OSession As OhioSession

Dim OFields As OhioFields

Dim OField As OhioField

. . .

Set OFields = OScreen.Fields

Set OPos = OSession.CreateOhioPosition(1, 1)

Set OField = OFields.FindByPosition(OPos)
```

C++ Example

```
IOhioSessions* pIOhioSessions;

IOhioFields* pIOhioFields;

IOhioField* pIOhioField;

IOhioPosition* inPos;

. . .

pIOhioSession->CreateOhioPosition

(24,1,&inPos);

HRESULT hr = pIOhioFields->FindByPosition

(inPos, &pIOhioField);

if (FAILED(hr)) {…} //error
```

Property: IOhioPosition::Column

This property returns or sets the column coordinate of the OhioPosition object.

Basic Syntax
Long = OhioPosition.**Column**
OhioPosition.**Column** = Long

C++ Syntax
HRESULT IOhioPosition::**get_Column**([out, retval] long * *pVal*);
HRESULT IOhioPosition::**put_Column**([in] long *newVal*);

Parameters
pVal—The returned column coordinate.
newVal—The value of the column coordinate.

Basic Example

```
Dim OSession As OhioSession

Dim OPos As OhioPosition

Dim iCol As Long

. . .

Set OPos = OSession.CreateOhioPosition(1, 1)

‘set column

OPos.Column = 5

‘get column

iCol = OPos.Column
```

C++ Example

```
long outCol, inCol;

IOhioSession* pIOhioSession;

IOhioPosition* OPos;

. . .

pIOhioSession->CreateOhioPosition

(5,8,&OPos);

//get the current Column

OPos->get_Column(&outCol);

//set the Column to another value

inCol = 14;

OPos->put_Column(inCol);
```

Property: IOhioPosition::Row

This property returns or sets the row coordinate of the OhioPosition object.

Basic Syntax

Long = OhioPosition.**Row**
OhioPosition.**Row** = Long

C++ Syntax

HRESULT IOhioPosition::**get_Row**([out, retval] long * *pVal*);
HRESULT IOhioPosition::**put_Row**([in] long *newVal*);

Parameters

pVal—The returned row coordinate.
newVal—The value of the row coordinate.

Basic Example

```
Dim OSession As OhioSession

Dim OPos As OhioPosition

Dim iRow As Long

. . .

Set OPos =

OSession.CreateOhioPosition(1, 1)

'set row

OPos.row = 5

'get row

iRow = OPos.Row
```

C++ Example

```
IOhioSession* pIOhioSession;

long outRow, inRow;

IOhioPosition* OPos;

. . .

pIOhioSession->CreateOhioPosition

(5,8,&OPos);

//get the current row value

OPos->get_Row(&outRow);

//set the row value

inRow = 14;

OPos->put_Row(inRow);
```

Data Types of OHIO

Ohio contains the following read-only data types:

Ohio interface

OHIO_DIRECTION

OhioSession interface

OHIO_STATE

OHIO_TYPE

OhioScreen interface

OHIO_COLOR

OHIO_EXTENDED

OHIO_FIELD

OHIO_PLANE

OHIO_UPDATE

OhioOIA interface

OHIO_INPUTINHIBITED

OHIO_OWNER

OHIO_COLOR Data Type

The OHIO_COLOR data type specifies the color of the text in the entire field.

It has the following values:

OHIO_COLOR_BLACK	Indicates that the color of the text is black.
OHIO_COLOR_BLUE	Indicates that the color of the text is blue.
OHIO_COLOR_GREEN	Indicates that the color of the text is green.
OHIO_COLOR_CYAN	Indicates that the color of the text is cyan.
OHIO_COLOR_RED	Indicates that the color of the text is red.
OHIO_COLOR_MAGENTA	Indicates that the color of the text is magenta.
OHIO_COLOR_WHITE	Indicates that the color of the text is white.
OHIO_COLOR_YELLOW	Indicates that the color of the text is yellow.

Related Topics

Method: IOhioField::GetData

Method: IOhioScreen::GetData

OHIO_DIRECTION Data Type

The OHIO_DIRECTION data type specifies the direction of the search.

It has the following values:

OHIO_DIRECTION_FORWARD	Indicates that the direction of the search is forward (from the beginning to the end).
OHIO_DIRECTION_BACKWARD	Indicates that the direction of the search is backward (from the end to the beginning).

Related Topics

Method: IOhioFields::FindByString

Method: IOhioScreen::FindString

OHIO_EXTENDED Data Type

The OHIO_EXTENDED data type specifies the extended attribute of the field.

It has the following values:

OHIO_EXTENDED_HILITE	Indicates the bitmask for highlighting bits.
OHIO_EXTENDED_COLOR	Indicates the bitmask for color bits.
OHIO_EXTENDED_RESERVED	Indicates the bitmask for reserved bits.

Related Topics

Method: IOhioField::GetData

Method: IOhioScreen::GetData

OHIO_EXTENDED_COLOR Data Type

The OHIO_EXTENDED_COLOR data type specifies the color of the individual character and overrides the OHIO_COLOR data type.

The OHIO_EXTENDED_COLOR data type has the following values:

OHIO_EXTENDED_COLOR_DEFAULT	Indicates that the color of the text is the default color.
OHIO_EXTENDED_COLOR_BLUE	Indicates that the color of the text is blue.
OHIO_EXTENDED_COLOR_RED	Indicates that the color of the text is red.
OHIO_EXTENDED_COLOR_PINK	Indicates that the color of the text is pink.
OHIO_EXTENDED_COLOR_GREEN	Indicates that the color of the text is green.
OHIO_EXTENDED_COLOR_TURQUOISE	Indicates that the color of the text is turquoise.
OHIO_EXTENDED_COLOR_YELLOW	Indicates that the color of the text is yellow.
OHIO_EXTENDED_COLOR_WHITE	Indicates that the color of the text is white.

Related Topics

Method: IOhioField::GetData

Method: IOhioScreen::GetData

OHIO_EXTENDED_HILITE Data Type

The OHIO_EXTENDED_HILITE data type specifies the type of the highlighted text.

It has the following values:

OHIO_EXTENDED_HILITE_NORMAL	Indicates normal highlighting.
OHIO_EXTENDED_HILITE_BLINK	Indicates that the highlighted text is flashing.
OHIO_EXTENDED_HILITE_ REVERSEVIDEO	Reverses the foreground and background color.
OHIO_EXTENDED_HILITE_ UNDERSCORE	Indicates that the field is underscored.

Related Topics

Method: IOhioField::GetData

Method: IOhioScreen::GetData

OHIO_FIELD Data Type

The OHIO_FIELD data type specifies the field type.

It has the following values:

OHIO_FIELD_ATTRIBUTE	Indicates that the byte in the data stream (buffer) contains a field attribute.
OHIO_FIELD_PROTECTED	Indicates that the field is not writable.
OHIO_FIELD_NUMERIC	Indicates that you can enter only numbers in the field.
OHIO_FIELD_PEN_SELECTABLE	Indicates that you can select the field.
OHIO_FIELD_HIGH_INTENSITY	Indicates that the field is highlighted, bold, and bright.
OHIO_FIELD_HIDDEN	Indicates that the field cannot be displayed.
OHIO_FIELD_RESERVED	Indicates that the field is reserved.
OHIO_FIELD_MODIFIED	Indicates that the field has been modified by a host.

Related Topics

Method: IOhioScreen::GetData

OHIO_INPUTINHIBITED Data Type

The OHIO_INPUTINHIBITED data type specifies what is inhibiting the input.

It has the following values:

OHIO_INPUTINHIBITED_NOTINHIBITED	Indicates that the input is not inhibited.
OHIO_INPUTINHIBITED_SYSTEM_WAIT	Indicates that the input is inhibited by a system wait state.
OHIO_INPUTINHIBITED_COMMCHECK	Indicates that the input is inhibited by a communications check state.
OHIO_INPUTINHIBITED_PROGCHECK	Indicates that the input is inhibited by a program check state.

OHIO_INPUTINHIBITED_MACHINECHECK	Indicates that the input is inhibited by a machine check state.
OHIO_INPUTINHIBITED_OTHER	Indicates that the input is inhibited by a state other than those listed above.

Related Topics

Property: IOhioOIA::InputInhibited

OHIO_OWNER Data Type

The OHIO_OWNER data type specifies the owner of the Ohio session.

It has the following values:

OHIO_OWNER_UNKNOWN	Indicates that the owner is unitialized.
OHIO_OWNER_APP	Indicates that the owner is an application or 5250 host.
OHIO_OWNER_MYJOB	Indicates that the owner is an application or 3270 host.
OHIO_OWNER_NVT	Indicates that the owner is a 3270 host (NVT or VT-XXX terminal).
OHIO_OWNER_UNOWNED	Indicates that the owner is a 3270 host (unowned).
OHIO_OWNER_SSCP	Indicates that the owner is a 3270 host (SSCP).

Related Topics

Property: IOhioOIA::Owner

OHIO_PLANE Data Type

The OHIO_PLANE data type specifies the plane from which to retrieve the data.

It has the following values:

OHIO_PLANE_TEXT	Indicates the text plane that contains character data.
OHIO_PLANE_COLOR	Indicates the color of each character in the particular field. This value uses the standard HLLAPI CGA color values.
OHIO_PLANE_FIELD	Returns the attribute of the field.
OHIO_PLANE_EXTENDED	Returns the extended attributes of the field. These attributes extend the function of OHIO_PLANE_FIELD.

Related Topics

Method: IOhioField::GetData

Method: IOhioScreen::GetData

OHIO_STATE Data Type

The OHIO_STATE data type specifies the status of the communication link to the host.

It has the following values:

OHIO_STATE_DISCONNECTED	Indicates that the communication link to the host is disconnected.
OHIO_STATE CONNECTED	Indicates that the communication link to the host is connected.

Related Topics

Method: IOhioSession::OnSessionChanged

OHIO_TYPE Data Type

The OHIO_TYPE data type specifies the type of host.

It has the following values:

OHIO_TYPE UNKNOWN	Indicates that the host type is unknown.
OHIO_TYPE_3270	Indicates that the host type is 3270.
OHIO_TYPE_5250	Indicates that the host type is 5250.
OHIO_TYPE_VT	Indicates that the host type is VT.

Related Topics

Property: IOhioSession::SessionType

OHIO_UPDATE Data Type

The OHIO_UPDATE data type specifies whether the host or client initiated the update of the screen.

It has the following values:

OHIO_UPDATE_HOST	Indicates that the host initiated the update.
OHIO_UPDATE_CLIENT	Indicates that the client initiated the update.

Related Topics

Method: IOhioScreen::OnScreenChanged

About Legacy APIs

HostExplorer provides the following existing or “legacy” APIs:

· OLE Automation—A Windows tool that lets you automate the exchange of data between applications, and lets you access and control HostExplorer.

· EHLLAPI (Extended High Level Language Application Programming Interface) and WinHLLAPI (Windows HLLAPI)—Allow other Windows programs (for example, Attachmate® Extra! for Windows) to communicate and control HostExplorer terminal emulators.

· DDE (Dynamic Data Exchange)—A tool that allows programs (for example, Microsoft Excel, Word, and Visual Basic) to communicate with the HostExplorer 3270 emulator.

While these APIs are less efficient and use larger and more rigid objects than COM and OHIO, you can still use them to write applications and thus avoid rewriting your own code. HostExplorer’s support of these earlier APIs helps maximize an organization’s investment in its development.

OLE Automation 3270 5250 VT

OLE Automation is a facility provided by Windows that lets you exchange data between applications. You can use OLE Automation to automate these tasks.

You can also use OLE Automation to access and control HostExplorer. You can write OLE Automation clients using a variety of tools, including Hummingbird Basic, Visual Basic, C++, and other languages.

The name of the Automation object is "HostExplorer."

The following example uses Visual Basic:

```
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
```

Now, use the language extensions as you would in the built-in macro editor

```
HE.CurrentHost.Keys "Login JOHN@E"
…
```

If you need to add OLE objects to a project during development, add the HOSTEX32.EXE executable file as a reference. Visual Basic then adds the objects, properties, and methods to your project. You can view the objects using the Object Browser.

Some sample macros are provided in the EB folder located in the main HostExplorer program folder.

Related Topics

OLE Objects

Methods and Properties of the Application Object

Methods and Properties of the Field Object

Methods and Properties of the Host Object

Methods and Properties of the Hosts Object

Methods and Properties of the Cfg3270, Cfg5250 and CfgVT Objects

OLE Objects 3270 5250 VT

HostExplorer provides methods and properties for the following OLE objects:

Application

Field

Host

Hosts

Cfg3270, Cfg5250, and CfgVT

A method is a construct that, when executed, performs an action and possibly returns a value. For example, you can use the Keys method to simulate pressing keys in HostExplorer. This method returns the value of the return code, indicating whether you pressed the keys successfully. A method can optionally take parameters.

A property is an interface to a variable in HostExplorer. Unlike a method, it does not perform an action or take any parameters. You can use a property to retrieve or set the value of its associated variable in HostExplorer. For example, you can use the AllowUpdates host property to get or set the value of the Screen Updates flag, which determines whether the program updates the screen. Properties that you can retrieve, but not alter, are called read-only properties.

Related Topics

OLE Automation

Methods and Properties of the Application Object 3270 5250 VT

The methods and properties of the Application object let you manipulate various aspects of a HostExplorer session.

The following methods and properties are available:

CurrentHost Property

DelSession Method

ExitAll Application Method

GetCurrentDir Method

GetFilePath Method

GetProfileString Method

Hosts Collection Object

NewSession Method

StartSession Method

Word Method

WriteProfileString Method

CurrentHost Property 3270 5250 VT

The CurrentHost property is a pointer to the current host object, which is the session with the last focus. If there is only one session, this object represents a pointer to that session.

Note: Because this object represents the session that last had the focus, the value of CurrentHost may change throughout your program when you start new sessions. To prevent this, use the Host object by assigning a static reference to the current session. For example:

```
Dim Host as Object
Set Host = HE.CurrentHost
… Now use Host.* object …
```

Syntax	CurrentHost
Group	Application Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Dim Host as Object Set HE = CreateObject("HostExplorer") Set Host = HE.CurrentHost Host.PrintScreen End Sub

Related Topics

Hosts Collection Object

Hosts Collection Object 3270 5250 VT

The Hosts Collection object is a collection of Host objects that represents all the screens in HostExplorer. Although it is not an array, you may think of it as a dynamic array. You must use this object to select a specific terminal session with which to communicate. The first session is session 0

Although the Hosts object contains its own set of properties, the particularly useful ones are the Hosts(n), and Hosts.Count properties. Hosts(n) returns a pointer to the *n*th session available. Hosts.Count returns the total number of sessions available.

To enumerate or find all active sessions, see "Enumerating Active Sessions". 

Syntax	Set Host = Hosts(index%) Hosts.Count Index%: index of session
Group	Application Object
Mode	3270, 5250, VT
Example	Sub Main Dim Host as Object Dim HE as Object Set HE = CreateObject("HostExplorer") Set Host = HE.Hosts(0) Num% = HE.Hosts.Count MsgBox "The total number of sessions available is " & Num% End Sub

Related Topics

CurrentHost Object

Enumerating Active Sessions 3270 5250 VT

You can use the following sample code to find active sessions:

```
Sub Main
Dim HE as Object
Dim Host as Object

Set HE = CreateObject( "HostExplorer" )

For i = 0 to 25
Set Host = HE.Hosts(i)

If Not Host Is Nothing Then
' Do whatever you need here...
End If
Next i
End Sub
```

DelSession Method 3270 5250 VT

You can use the DelSession method to delete a session created with the NewSession method but only when you have never started and used the session. If you create a new session with NewSession and then connect to a host, the system deletes the session after the window closes. The first available session has index 0. Use the Close host method to close a session properly.

Note: Do not use this method to terminate an existing session. This method does not terminate network links. Disconnect the session before deleting it.

Syntax	DelSession index% Index%: Index of session to delete
Group	Application Object
Mode	3270, 5250, VT
Example	Sub Main Dim Host as Object Dim HE as Object Set HE = CreateObject("HostExplorer") index% = HE.NewSession Set Host = HE.Hosts(index%) … some work … HE.CurrentHost.Disconnect HE.DelSession index% End Sub

Related Topics

NewSession Method

StartSession Method

NewSession Method 3270 5250 VT

You can use the NewSession method to create a new session control block within HostExplorer. You use this method if you want to create a new session from scratch without using a previously saved profile. NewSession returns a value representing the index number of the new session. The first available session has index 0.

Syntax index% = NewSession

Group Application Object

Mode 3270, 5250, VT

Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
Dim Host as Object
Dim HE as Object
Dim Cfg as Object
Set HE = CreateObject( "HostExplorer" )

index% = HE.NewSession
Set Host = HE.Hosts(index%)
Set Cfg = Host.CfgVT ' Use CfgVT,

Cfg3270, or Cfg5250
Host.TerminalMode =TERMINAL_TELNET '

Set terminal mode first!
Host.Model =TELNET_VT220 ' Set terminal

model next
Host.ConnectBy = IO_TELNET ' Set

transport next

Cfg.Host = "myhost.mydomain"
Cfg.ConnectTimeout = 30
Cfg.TCPPort = 23

Host.Connect
End Sub
```

Related Topics

StartSession Method

StartSession Method 3270 5250 VT

You can use the StartSession method to start a new session from a profile saved on disk. This method takes one parameter that is the name of the profile. You can specify the ProfileName in one of two ways:

· Specify a complete directory path to the MyProfile.HEP file in the Profile\3270 directory where the user files are stored on your machine. For the appropriate directory path for your platform, refer to the list of the default locations for the user files. 

· Specify the older format if the profile is in a first level folder "MyProfile.MyFolder".

The value index% returned is the index for the host session created. You can use this index to access the Host object directly.

Syntax

index% = StartSession(szProfileName$)

szProfileName$: name of profile to load and connect

index%: index of session started

Group Application Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Dim Host as Object
Set HE = CreateObject( "HostExplorer" )

index% = HE.StartSession( "VMCMS.3270" )

Set Host = HE.Hosts( index% )
End Sub
```

Related Topics

NewSession Method

Default Locations for User Files

Hummingbird Setup Wizard installs per-user files (current user and "All Users") to various directories depending on the Windows platform.

For Windows NT/2000 platforms, the wizard prompts you to choose between installing the product on the computer for the currently logged in user, or for all users. For the current user, shortcuts are created in the appropriate user profile folder, along with copies of all other user files. For all users (anyone who uses the computer), shortcuts are created in the "All Users" profile folder, along with copies of all other user files.

The following are the default locations for user files:

Operating System	Per-User Files—Default Location (Current User)
Windows 95/98/Me	C:\Windows\Application Data
Windows 95/98/Me (user profiles enabled)	C:\Windows\Profiles\ %USERNAME%\Application Data
Windows NT 4.0	C:\Winnt\Profiles\ %USERNAME%\Application Data
Windows 2000	C:\Documents and Settings\%USERNAME%\ Application Data (hidden)

Operating System	Per-User Files—Default Location (All Users)
Windows 95/98/Me	C:\Windows\Application Data
Windows 95/98/Me (user profiles enabled)	C:\Windows\Profiles\ All Users\Application Data
Windows NT 4.0	C:\Winnt\Profiles\All Users\Application Data
Windows 2000	C:\Documents and Settings\All Users\Application Data (hidden)

ExitAll Application Method 3270 5250 VT

You can use this method to immediately terminate and close all active sessions. Make this method the last call to HostExplorer because it automatically unloads the emulator.

Syntax	ExitAll
Group	Application Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") … some work … HE.ExitAll End Sub

Related Topics

Close Host Method

Close Host Method 3270 5250 VT

You can use this method to close (terminate) the host session and object. This method immediately terminates any host connection and closes the session window.

Syntax Close

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

' Close the current host session
HE.CurrentHost.Close
End Sub
```

GetCurrentDir Method 3270 5250 VT

The GetCurrentDir method returns the current working directory.

Syntax

```
szDir$ = GetCurrentDir
```

szDir$: current working directory

Group Application Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

Path$ = HE.GetCurrentDir
Print Path$
End Sub
```

GetFilePath Method 3270 5250 VT

The GetFilePath method displays a dialog box that prompts you for a file path. The returned string is a complete path and file name. If you click the Cancel button, the system returns a null string.

Syntax

szPath$ = GetFilePath ([szDefName$], [szDefExt$], [szDefDir$], [szTitle$], [iOption%])

szDefName$: used to set the initial file name. If you omit this, *.szDefExt$ is used.

szDefExt$: used to initially show files whose extension matches this string value. You can specify multiple extensions by using a comma (,) as the separator. If you omit this, * is used.

szDefDir$: used to set the initial directory. If you omit this, the system used the current directory. You can useTitle$ to set the title of the dialog box. If you omit this, "Open" is used.

iOption%: numeric value which determines the file selection options. If you omit this, the system uses a zero.

The available options are:

0 - Only allow the user to select a file that exists

1 - Confirm creation when the user selects a file that does not exist

2 - Allow the user to select any file whether it exists or not

3 - Confirm overwrite when the user selects a file that exists

szPath$: path string returned from the dialog box if the user pressed OK.

Selecting a different directory changes the application's current directory.

Group Application Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

Print HE.GetFilePath( "Hostex32.exe",
"EXE",
"Hostex32.exe"
"Program Files"
"Get Directory",0)
End Sub
```

GetProfileString Method 3270 5250 VT

You can use the GetProfileString method to retrieve a value from any INI file.

The szSection$ parameter is the section name. This parameter must always be present. The szKey$ parameter is the key name. You can pass an empty string as this parameter to read all the keys in the section. The system separates all keys by newline characters. The szINIFileName$ parameter is the file name. If you omit a path, the system looks for the file in the Windows directory.

Syntax

szValue$ = GetProfileString(szSection$, szKey$, szINIFileName$)

szSection$: section of the file (the name is square brackets)

szKey$: key to read

szINIFileName$: name of the INI file to read

szValue$: string value read from the INI file

Group Application Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szBeep$ = HE.GetProfileString( "Windows",

"Beep", "WIN.INI" )
MsgBox "The content of that key is: " &

szBeep$, , "Output"
End Sub
```

Related Topics

WriteProfileString Method

WriteProfileString Method 3270 5250 VT

You can use the WriteProfileString method to write an INI key value to a file. The szSection$ parameter is the section name and is mandatory. The szKey$ parameter is the key name. If you pass an empty string as the szKey$ and szValue$ parameters, the entire section is deleted. The szValue$ parameter is the value for the key. If you pass an empty string as the szValue$, then the key is deleted from the section. The szINIFileName$ parameter is the file name. If you omit a path, the system looks for the file in the Windows directory.

Syntax

WriteProfileString szSection$, szKey$, szValue$, szINIFileName$

szSection$: section of the file (the name In square brackets)

szKey$: key to write

szINIFileName$: is the name of the INI file to write

Group Application Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.WriteProfileString "Windows", "Beep",

"No", "WIN.INI"
End Sub
```

Related Topics

GetProfileString Method

Word Method 3270 5250 VT

You can use the Word method to parse out the nth word of a string. The first parameter is the string that you want to parse. The second parameter contains a string of valid delimiters to parse the words. The third parameter is the index of the word to retrieve. Unlike other methods, the first word of a string has an index of 1.

Syntax

szWord$ = Word(szString$, szParse$, index%)

szString$: string to parse

szParse$: string of delimiters to parse szString$

index%: index of word to parse out

szWord$: word parsed by method

Group Application Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szWord$ = HE.Word("I have a lazy brown

dog", " ", 2 )
MsgBox "The second word in the string is

" & """ & szWord$ & """

End Sub
```

Methods and Properties of the Field Object 3270 5250 VT

The field object methods and properties let you manipulate all aspects of a field. You can access field methods directly, using the With statement or by creating an object for easy reference.

Example:

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
HE.CurrentHost.Fields(7).Text = "Hello World"
End Sub
```

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
With HE.CurrentHost.Fields(7)
.Text = "Hello World"
End With
End Sub
```

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
Dim Fld as Object
Set Fld = HE.CurrentHost.Fields(7)
Fld.Text = "Hello World"
End Sub
```

The following field object methods and properties are available:

Hosts.Fields Collection Object

Fields Collection Object

Attr Field Property

ExtAttr Field Property

Length Field Property

IsBold Field Property

IsHidden Field Property

IsModified Field Property

IsNumeric Field Property

IsPenSelectable Field Property

IsProtected Field Property

Pos Field Property

Text Field Property

Hosts.Fields Collection Object 3270 5250

The Hosts.Fields Collection Object is a collection of Field objects. It represents all fields on the current host screen in a logical fashion. Although it is not an array, you may think of it as a dynamic array. Although the object contains its own set of properties, the particularly useful ones are the Fields(n), and Fields.Count properties. The Fields(n) return a pointer to the *n*th field available. The Field index starts at 0. Fields.Count returns the total number of fields.

In 3270 mode, fields can be protected or unprotected. In a screen that contains at least one attribute, you can typically represent the entire screen by a collection of fields.

In 5250 mode, fields can only be unprotected. They represent the entries in the format table, areas in which you can type. You cannot use a field object to get data from a protected portion of the screen.

Syntax

```
Dim Fld as Object
 Set Fld = CurrentHost.Fields(n)
 iNumFields = CurrentHost.Fields.Count
```

Group Field Object

Mode 3270, 5250

Example

```
Sub Main
Dim Fld as Object
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
Set Fld = HE.CurrentHost.Fields(2)

MsgBox "Value of 2'nd field is: " + Fld.Text
End Sub
```

Related Topics

FieldID Host Method

FieldID Host Method 3270 5250

This method returns the field index for a given position on the screen. You can use this method to access the field object directly. The first position on the screen is 1, whereas the first Field ID index is 0.

Syntax

iFieldID% = FieldID(iPos%)

iPos%: position to query

iFieldID%: field ID returned for queried position

Group Host Object

Mode 3270, 5250

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iFldID = HE.CurrentHost.FieldID( 1760 )
MsgBox "The ID of the field at position 1760 is " & """ + iFldID & """, ,
"Information"
End Sub
```

Related Topics

Methods and Properties of the Field Object

Fields Collection Object 3270 5250

The Fields Collection object lets you manipulate a field through a series of methods and properties. This interface provides a device-independent way of dealing with data on the host screen. The first field is Field 0.

Syntax

```
Dim Fld as Object
Set Fld = CurrentHost.Fields( iFieldNum% )

iFieldNum%: field index to return
```

Group Host Object

Mode 3270, 5250

Example

```
Sub Main
Dim Fld as Object
Dim HE as Object

Set HE = CreateObject( "HostExplorer" )
Set Fld = HE.CurrentHost.Fields(2)

MsgBox "Field 2 has length: " + Str$(Fld.Length) + ", Position: " + Str$(Fld.Pos) + ",
Value: " + Fld.Text
End Sub
```

Related Topics

Methods and Properties of the Field Object

Attr Field Property 3270 5250

Within TN3270, this read-only property returns the 3270 attribute of the field selected by the Field object. You can use the following literals to test the bit options:

· ATTR_MODIFIED

· ATTR_BOLD

· ATTR_NUMERIC

· ATTR_PROTECTED

· ATTR_NONDISPLAY

Within TN5250, this read-only property returns the Field Format Word of the field selected by the Field object.

You can use the following literals to test the bit options:

· FFW_BYPASS

· FFW_ALLOWDUP

· FFW_MDT

· FFW_SHIFTEDITMASK

· FFW_ALPHASHIFT

· FFW_ALPHAONLY

· FFW_NUMERICSHIFT

· FFW_NUMERICONLY

· FFW_KATASHIFT

· FFW_DIGITSONLY

· FFW_IO

· FFW_SIGNEDNUMERIC

· FFW_AUTOENTERONEXIT

· FFW_FIELDEXITREQUIRED

· FFW_MONOCASE

· FFW_MANDATORYENTRY

· FFW_RIGHTFILLMASK

· FFW_NOADJUST

· FFW_RIGHTADJUSTZEROFILL

· FFW_RIGHTADJUSTBLANKFILL

· FFW_MANDATORYFILL

Syntax	iAttr% = Attr
Group	Field Object
Mode	3270, 5250

Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
Dim Fld as Object
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
' Get second field object
Set Fld = HE.CurrentHost.Fields(2)

MsgBox "Field 2 has an attribute of: " + Str$(Fld.Attr)
End Sub
```

Related Topics

ExtAttr Field Property

ExtAttr Field Property 3270 5250

Within TN3270, this read-only property returns the extended 3270 attribute of the field selected by the Field object. The extended attribute defines the color and highlighting for the field. You can use the following literals to test the bit options:

· ATTR_REVERSE

· ATTR_BLINK

· ATTR_UNDERLINE

Within TN5250, this read-only property returns the Field Control Word of the field selected by the Field object.

You can use the following literals to test the bit options:

· FCW_LIGHTPEN

· FCW_STRIPEANDLIGHTPEN

Syntax	iExtAttr% = ExtAttr
Group	Field Object
Mode	3270, 5250
Example	'$Include:"-E\hebasic.ebh" Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field 2 has an extended attribute of: " + Str$(Fld.ExtAttr) End Sub

Related Topics

Attr Field Property

Length Field Property 3270 5250

This read-only property returns the length of the field selected by the Field object.

Syntax iLen% = Length

Group Field Object

Mode 3270, 5250

Example

```
Sub Main
Dim Fld as Object
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
' Get second field object
Set Fld = HE.CurrentHost.Fields(2)

MsgBox "Field 2 has length: " + Str$(Fld.Length)
End Sub
```

IsBold Field Property 3270

This read-only property returns TRUE (-1) if the field is bold or FALSE (0) if it is not.

Syntax	bVal = IsBold
Group	Field Object
Mode	3270
Example	Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field is Bold " + Fld.IsBold End Sub

Related Topics

IsHidden Field Method

IsModified Field Method

IsNumeric Field Method

IsPenSelectable Field Method

IsProtected Field Method

IsHidden Field Property 3270 5250

This read-only property returns TRUE (-1) if the field is hidden (non-visible) or FALSE (0) if it is not (visible).

Syntax	bVal = IsHidden
Group	Field Object
Mode	3270, 5250
Example	Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field is Hidden " + Fld.IsHidden End Sub

Related Topics

IsBold Field Method

IsModified Field Method

IsNumeric Field Method

IsPenSelectable Field Method

IsProtected Field Method

IsModified Field Property 3270

This read-only property returns TRUE (-1) if the field is modified or FALSE (0) if it is not.

Syntax	bVal = IsModified
Group	Field Object
Mode	3270
Example	Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field is Modified " + Fld.IsModified End Sub

Related Topics

IsBold Field Method

IsHidden Field Method

IsNumeric Field Method

IsPenSelectable Field Method

IsProtected Field Method

IsNumeric Field Property 3270

This read-only property returns TRUE (-1) if the field is numeric or FALSE (0) if it is not.

Syntax	bVal = IsNumeric
Group	Field Object
Mode	3270
Example	Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field is Numeric " + Fld.IsNumeric End Sub

Related Topics

IsBold Field Method

IsHidden Field Method

IsModified Field Method

IsPenSelectable Field Method

IsProtected Field Method

IsPenSelectable Field Property 3270

This read-only property returns TRUE (-1) if the field is pen selectable or FALSE (0) if it is not.

Syntax	bVal = IsPenSelectable
Group	Field Object
Mode	3270
Example	Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field is PenSelectable " + Fld.IsPenSelectable End Sub

Related Topics

IsBold Field Method

IsHidden Field Method

IsModified Field Method

IsNumeric Field Method

IsProtected Field Method

IsProtected Field Property 3270

This read-only property returns TRUE (-1) if the field is protected or FALSE (0) if it is not.

Syntax	bVal = IsProtected
Group	Field Object
Mode	3270
Example	Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field is Protected " + Fld.IsProtected End Sub

Related Topics

IsBold Field Method

IsHidden Field Method

IsModified Field Method

IsNumeric Field Method

IsPenSelectable Field Method

Pos Field Property 3270 5250

This read-only property returns the position of the field selected by the Field object. The first position on the screen is 1.

Syntax	iPos% = Pos
Group	Field Object
Mode	3270, 5250
Example	Sub Main Dim Fld as Object Dim HE as Object Set HE = CreateObject("HostExplorer") ' Get second field object Set Fld = HE.CurrentHost.Fields(2) MsgBox "Field 2 has position: " + Str$(Fld.Pos) End Sub

Related Topics

Length Field Property

Text Field Property 3270 5250

You can use this property to get or set the contents of the field as a string. The system converts all Nulls to Blanks and removes all trailing blanks.

Syntax szText$ = Text

Group Field Object

Mode 3270, 5250

Example

```
Sub Main
Dim Fld as Object
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
' Get second field object
Set Fld = HE.CurrentHost.Fields(2)

MsgBox "Field 2 has value: " + Fld.Text
End Sub
```

Methods and Properties of the Host Object 3270 5250 VT

The Host Object methods and properties let you configure and manipulate all aspects of a screen. You can access a host screen in two ways. You can use the CurrentHost object, which always refers to the screen which has or last had the focus, or the Hosts(n) object where *n* is a value from 1 to Hosts.Count (the total number of available sessions). The latter is the method used to access the n'th session available. If you are writing scripts that simply access one session, use of the CurrentHost object is the recommended method. You can access host methods directly using the With statement, or by creating an object for easy reference.

The following Host Object methods and properties are available:

Host Activate Method 3270 5250 VT

You can use this method to activate the host object. This brings the host object to focus by making the screen window the top-level window on the desktop.

Syntax	Activate
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Bring this window to the top HE.CurrentHost.Activate End Sub

Related Topics

Hide Host Method

Show Host Method

Hide Host Method 3270 5250 VT

You can use this method to hide the host object. This will hide the screen whether it is minimized, normal, or maximized. When the screen is hidden, it no longer appears in the taskbar.

Syntax	Hide
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Hide the current host screen HE.CurrentHost.Hide End Sub

Related Topics

Activate Host Method

Show Host Method

Show Host Method 3270 5250 VT

You can use this method to show the Host object. This displays the screen whether it is minimized, normal, or maximized.

Syntax	Show
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Show the current host screen HE.CurrentHost.Show End Sub

Related Topics

Hide Host Method

OIAUpdated Host Property 3270 5250 VT

If the system returns a TRUE value (Updated), and the operator information area (OIA) was updated since the last call, the system will reset the internal flag to FALSE (Not Updated) after the call. This is a read-only property.

Syntax	bVal = OIAUpdated
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bUpdated = HE.CurrentHost.OIAUpdated MsgBox "OIA Return Value is " & bUpdated, , "Information" End Sub

Related Topics

OIA Host Property

OIA Host Property 3270 5250 VT

This property returns the operator information area (OIA) as a text string. This is a read-only property.

Syntax	szOIA$ = OIA
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szOIA = HE.CurrentHost.OIA MsgBox szOIA, , "Content of Operator Information Area (OIA)" End Sub

Related Topics

OIAUpdated Host Property

AllowClose Host Property 3270 5250 VT

You can use this property to allow, or to disallow the user from closing the current session. To prevent the session from closing, set the AllowClose property to FALSE. To allow the user to close the session, set the AllowClose property to TRUE.

Syntax	AllowClose = TRUE/FALSE
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bAllowClose = HE.CurrentHost.AllowClose HE.CurrentHost.AllowClose = FALSE End Sub

Related Topics

AllowUpdates Host Property

QueryCloseRequest Host Property

AllowUpdates Host Property 3270 5250 VT

You can use this property to return or set the screen updates flag. When disabled, the system does not perform screen updates. Use this flag to optimize performance or prevent a user from seeing updates while you are performing transactions.

Syntax	AllowUpdates = TRUE/FALSE
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bUpdates = HE.CurrentHost.AllowUpdates HE.CurrentHost.AllowUpdates = FALSE End Sub

Related Topics

AllowClose Host Property

PSUpdated Host Property

PSUpdated Host Property 3270 5250 VT

This read-only property returns a TRUE value if the presentation space was updated since the last call to this property. This property resets the internal flag to FALSE (not updated) after each call.

Use this function to synchronize with host applications.

Syntax	bVal = PSUpdated
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bUpdated = HE.CurrentHost.PSUpdated MsgBox "Presentation Space Value is " & bUpdated, , "Information" End Sub

Related Topics

AllowUpdates Host Property

QueryCloseRequest Host Property 3270 5250 VT

You can use this property to see if the user attempted to close the session. The R/O property returns TRUE if the user attempted to close the session (File/Close Session, File/Exit All, Alt-F4). This query is valid only if you disabled the session close using the AllowClose property.

Syntax	bCloseRequested = QueryCloseRequest
Mode	3270, 5250, Vt
Group	This property has no group.
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplore") HE.CurrentHost.AllowClose = FALSE bCloseRequest = HE.CurrentHost.QueryCloseRequest End Sub

Related Topics

AllowClose Host Property

Pause Host Method 3270 5250 VT

You can use this method to pause the macro for the specified time in milliseconds.

Syntax	PrintScreen
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Pause for 5 seconds HE.CurrentHost.Pause 5000 End Sub

PrintScreen Host Method 3270 5250 VT

You can use this method to print the current host session to the Windows printer specified in the profile for the session.

Syntax PrintScreen

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.PrintScreen
End Sub
```

BFPress Host Method 3270

You can use this method to send the modified data to the host system. The parameter allows you to specify the actual AID (Attention Identifier) sent to the host system in the 3270 datastream. This method is used to support RPQ 7J0048.

Syntax

BFPress iAIDValue%

iAIDValue%: AID key value for host datastream

Group

Host Object

Mode

3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.BFPress &h7D ' Press Enter key
End Sub
```

Related Topics

BFStatus Host Method

BFStatus Host Property 3270

This read-only property is used to return the status of the extended function keys used to support RPQ 7J0048. The string returned contains the status for each extended function key. There are 48 function keys. The status character for each key can be: '0' - Key is Off, '1' - Key is On, '2' - Key is flashing. Therefore, to check the status of BF Key 10, you would check the tenth character in the string.

Syntax

```
szStatus$ = BFStatus

szStatus$: status string for all BF keys
```

Group Host Object

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

status$ = HE.CurrentHost.BFStatus
End Sub
```

Related Topics

BFPress Host Method

ProtectedText Host Method 3270 5250

You can use this method to replace text in protected locations of the screen. Use this method carefully since you can overwrite field attributes.

Syntax

ProtectedText iRow%, iCol%, szText$

iRow%: row to write text

iCol%: column to write text

szText$: text to write

Group Host Object

Mode 3270, 5250

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.ProtectedText 2, 2, "Hello World"
End Sub
```

PSReserved Host Property 3270 5250 VT

You can use this property to get or set the session-reserved flag. When you reserve a session (TRUE), the system does not allow user input. When the flag is FALSE (default), the user can type and use the menu. This makes the screen read-only.

Syntax	PSReserved = TRUE/FALSE
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bReserved = HE.CurrentHost.PSReserved HE.CurrentHost.PSReserved = FALSE End Sub

Related Topics

AllowClose Host Property

AllowUpdates Host Property

Bytes Host Property 3270 5250 VT

This read-only property returns the number of bytes in the presentation space of the host object. It is equal to the number of columns multiplied by the number of rows.

Syntax	iBytes% = Bytes
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iBytes = HE.CurrentHost.Bytes MsgBox "The number of Bytes in the presentation space is " & iBytes, , "Information" End Sub

Related Topics

Columns Host Property

Rows Host Property

Columns Host Property 3270 5250 VT

This read-only property returns the number of columns currently defined in the Host object.

Syntax	iCols% = Columns
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iColumns = HE.CurrentHost.Columns MsgBox "The number of Columns in presentation space is " & iColumns, , "Information" End Sub

Related Topics

Bytes Host Method

Rows Host Method

Rows Host Property 3270 5250 VT

This read-only property returns the number of rows currently defined in the host object.

Syntax	Rows
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iRows = HE.CurrentHost.Rows MsgBox "The number of rows in the presentation space is " & iRows & "Information" End Sub

Related Topics

Columns Host Method

Capture Host Property 3270 5250 VT

You can use this read/write property to get or set the capture mode for the host object. You can retrieve the current setting for the capture mode or you can set a new value.

You can enable constant capturing of TN3270 and TN5250 panels as they are received from the host. In Telnet, you can capture data in Text or Raw Mode. The capture file is set using the SaveFile Cfgxxxx property. 

Syntax	Capture = TRUE/FALSE
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bMode = HE.CurrentHost.Capture HE.CurrentHost.Capture = TRUE End Sub

Related Topics

SaveScreen Host Method

SaveFile Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Save Filename option.

Syntax SaveFile = szSaveFile$

Group Cfg3270, Cfg5250, and CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue = HE.CurrentHost.Cfg3270.SaveFile
HE.CurrentHost.Cfg3270.SaveFile = "C:\SAVED.TXT"
End Sub
```

SaveScreen Host Method 3270 5250 VT

You can use this method to save the session screen to a file.

Syntax	SaveScreen szFileName$ szFileName$: filename to save screen
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.SaveScreen "C:\SCREEN.TXT" End Sub

Related Topics

CaptureOIA Host Property

CaptureOIA Host Property 3270 5250

You can use this read/write property to get or set the capture operator information area (OIA) flag for the host object. When the value is TRUE, the system captures the OIA in every screen. When the value is FALSE, the system does not capture the OIA.

Syntax	CaptureOIA = TRUE/FALSE
Group	Host Object
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bMode = HE.CurrentHost.CaptureOIA HE.CurrentHost.CaptureOIA = TRUE HE.CurrentHost.SaveScreen "C:\SCREEN.TXT" End Sub

Related Topics

SaveScreen Host Method

PutText Method 3270 5250 VT

You can use this method to write text to any unprotected location on the screen. In VT mode, the iRow and iCol parameters are ignored and can be omitted. The first location on the screen is Row 1 Column 1. This method is identical to the Keys method but it allows you to specify the location to write the text.

Syntax	PutText szText$, iRow%, iCol% szText$: text to write iRow%: row to write text iCol%: column to write text
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.PutText "Hello World", 2, 2, End Sub

Related Topics

Keys Host Method

RunCmd Host Method

Keys Host Method 3270 5250 VT

You can use this method to press keys while in HostExplorer. You can insert special sequences in 3270, 5250, or VT mode to press system keys such as Tab, Reset, and so on. Valid return codes are:

· 0—String processed successfully

· 5—Error processing string. Keyboard locked.

Note: The Keys method is affected by the Type Ahead setting. If you send a 3270/5250 AID key and Type Ahead is enabled, the function will not return until the keyboard is unlocked by the host system. If Type Ahead is disabled, the function will always return immediately.

Syntax	iRc% = Keys(keystring$) keyString$: string to type iRc%: return code
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Send USERID, Tab, PASSWORD, Enter HE.CurrentHost.Keys "USERID@TPASSWORD@E" End Sub

Related Topics

3270/5250 Special Sequences

VT Special Sequences

PutText Host Property

RunCmd Host Method

3270/5250 Special Sequences

The format of the string is identical to the one used in the EHLLAPI, DDE, and VB interfaces. Listed below are special-character combinations. Keep in mind that they are case-sensitive.This method returns 0 if all keys were processed successfully. The Keys method is not the most efficient method of transferring large amounts of information to the screen buffer. For faster access, use the Fields.Text property.

@B Backtab

@C Clear

@D Delete

@E Enter

@F Erase EOF

@H 5250 Help

@I Insert

@J Next-Session

@L Cursor Left

@N Newline

@P 5250Print

@R Reset

@T Tab

@U Cursor Up

@V Cursor Down

@Z Cursor Right

@0 Home

@< Backspace

@1 PF1

@2 PF2

@3 PF3

@4 PF4

@5 PF5

@6 PF6

@7 PF7

@8 PF8

@9 PF9

@a PF10

@b PF11

@c PF12

@d PF13

@e PF14

@f PF15

@g PF16

@h PF17

@i PF18

@j PF19

@k PF20

@l PF21

@m PF22

@n PF23

@o PF24

@u Roll Up

@v Roll Down

@x PA1

@y PA2

@z PA3

@A@E Field Exit

@A@F Erase Input

@A@H Test Request

@A@J Cursor Select

@A@L Fast Left

@A@Q Attention

@A@- Field Minus

@A@+ Field Plus

@A@< Record Backspace

@A@Z Fast Right

@A@t Print Screen

@A@y Next Word

@A@z Prev Word

@S@x Duplicate

@S@y Field Mark

Related Topics

VT Special Sequences

VT Special Sequences

VT mode string formats are different to allow for special characters such as control characters and escape sequences. Enter Escape and binary codes in C-style syntax using the backslash character (\). The system treats in-line spaces as part of the sequence.

The sequence \xhh lets you specify any ASCII character as a hexadecimal character code. For example, you can give the ASCII backspace character as the normal C escape sequence (\b), or you can code it as \x08 hexadecimal.

You must use at least one digit for a hexadecimal escape sequence, but you can omit the second digit. Therefore, you can specify the hexadecimal escape sequence for the backspace as either \x8 or \x08.

Related Topics

3270/5250 Terminal Modes

Entering Control Sequences

Entering Control Sequences VT

The system treats the following in-line spaces as part of a sequence:

· \a—Bell (alert)

· \b—Backspace

· \e—Escape

· \f—Formfeed

· \n—Newline

· \r—Carriage Return

· \t—Horizontal Tab

· \v—Vertical Tab

· \'—Single quotation mark

· \"—Double quotation mark

· \\—Backslash

· \xhh — ASCII character in hexadecimal notation

When you type control sequences, you can use caret format. For example, to type a Ctrl-A value, you would type ^A. To type a caret, type the caret character twice, for example, (^^).

Example:

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

' Press tab key twice
HE.CurrentHost.Keys "\t\tTab Twice"

' Press Ctrl-C
HE.CurrentHost.Keys "^C"
End Sub
```

RunCmd Host Method 3270 5250 VT

You can use this method to run a system command within the session. You can assign system commands to keys, tool items and poppads. You can also use system commands within a macro.

Note: The RunCmd method is affected by the Type Ahead setting. If you send a 3270/5250 AID key (for example, Enter, PFx, and so on) and Type Ahead is enabled, the function will not return until the keyboard is unlocked by the host system. If Type Ahead is disabled, the function will always return immediately.

Syntax	RunCmd szCommandName$ szCommandName$: name of command to execute
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Show the Session Profile dialog HE.CurrentHost.RunCmd "Dlg-Edit-Session-Profile" End Sub

Related Topics

Keys Host Method

QuickKeyFile Host Property 3270 5250 VT

You can use this read-only property to return the set of Quick-Keys that are currently loaded.

Syntax	szQuickKey$ = QuickKeyFile
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szQuickKeys$ = HE.CurrentHost.QuickKeyFile End Sub

Related Topics

LoadQuickKeyFile Host Property

RunQuickKey Host Method

SaveQuickKeyFile Host Method

LoadQuickKeyFile Host Method 3270 5250 VT

You can use this method to load a new set of Quick-Keys for this session. Quick-Keys are shared between sessions and loading a new set of Quick-Keys from one session will affect other sessions. Valid return codes are:

· 0—Successfully loaded Quick-Keys.

· 1—Error loading Quick-Keys.

Syntax	iRc% = LoadQuickKeyFile(szFileName$) szFileName$: Quick-Key file to load. No path or extension specified. iRc%: return code for method
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iRc% = HE.CurrentHost.LoadQuickKeyFile("MYQUICKKEYS") End Sub

Related Topics

QuickKeyFile Host Property

RunQuickKey Host Method

SaveQuickKeyFile Host Method

RunQuickKey Host Method 3270 5250 VT

You can use this method to run a Quick-Key within the session.

Syntax	RunQuickKey szQuickKeyName$ szQuickKeyName$: name of Quick-Key to execute
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") 'Make sure you create a quick-key before running this macro HE.CurrentHost.RunQuickKey "dothis" End Sub

Related Topics

LoadQuickKeyFile Host Property

QuickKeyFile Host Property

SaveQuickKeyFile Host Method

SaveQuickKeyFile Host Method 3270 5250 VT

You can use this method to save Quick-Keys to disk. Do not supply a path or extension for the filename.

Syntax	Save QuickKeyFile szQuickKeyFileName$ szQuickKeyFileName$: file name of Quick-Keys to save
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.SaveQuickKeyFile "VMCMS" End Sub

Related Topics

LoadQuickKeyFile Host Property

QuickKeyFile Host Property

RunQuickKey Host Method

ReceiveFile Host Method 3270 VT

You can use this method to initiate a file transfer (download) from the host system.

In 3270 mode, the first parameter specifies the local file to receive the data, the second parameter specifies the host filename to download, and the last parameter specifies the host. 

In VT mode, the first parameter specifies the local file to receive the data and the second parameter specifies the file transfer protocol to use. When using ZModem, this parameter should be a path since the filename will be provided by the ZModem protocol.

Syntax	3270 ReceiveFile pcfilename$, hostfilename$, options$ VT ReceiveFile pcfilename$, iXferProtocol% pcfilename$: PC filename to receive host file hostfilename$: Host filename to download options$: options for this file transfer iXferProtocol%: File transfer protocol to use. Refer to HEBASIC.EBH for values.
Group	Host Object
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' 3270 mode download HE.CurrentHost.ReceiveFile "C:\PROFILE.TXT", "PROFILE EXEC A1", "(ASCII CRLF" HE.CurrentHost.WaitXfer ' VT mode download HE.CurrentHost.ReceiveFile "C:\VTFILE.TXT", XFER_TYPE_ZMODEM HE.CurrentHost.WaitXfer End Sub

Related Topics

IsXfer Host Property

SendFile Host Method

WaitXfer Host Method

XferCount Host Property

XferRC Host Property

Differences between Hummingbird Basic and WinWrap Basic

This section lists the differences between Hummingbird Basic and WinWrap Basic.

Note: WinWrap Basic was used in HostExplorer v4.x.

Differences between Hummingbird Basic and WinWrap:

Assignment Group

Erase—Reinitializes the elements of fixed-size arrays and releases dynamic storage space. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Let—Assign the value of an expression to a variable or property. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Lset—Left aligns a string variable or copies a variable of one user-defined type to another variable of a different user-defined type.

Set—Assign an object reference to a variable or property.

Erase—Reinitializes the contents of a fixed array and frees the storage associated with a dynamic array.

Let—Assigns the value of an expression to a variable or property.

Lset—Left aligns a string variable or copies a variable of one user-defined type to another variable of a different user-defined type.

New—In the Set statement, New allocates and initializes a new object of the named class.

Rset—Right aligns a string within a string variable.

Set—Assigns an object reference to a variable or property.

Constant Group

Empty—A variant that does not have any value. Not supported by Hummingbird Basic.

FALSE—An expression is false when its value is zero. Not supported by Hummingbird Basic, instead an integer value zero is used.

Nothing—An object value that does not refer to an object.

Null—The Null method returns a variant value set to the null value. Null is used to explicitly set a variant to the null value.

TRUE—An expression is TRUE when its value is non-zero. Not supported by Hummingbird Basic, instead an integer value not equal to zero is used.

Win16—TRUE if running in 16-bits. FALSE if running in 32-bits. Not supported by Hummingbird Basic.

Win32— TRUE if running in 32-bits. FALSE if running in 16-bits. Not supported by Hummingbird Basic.

Conversion Group

Array—Returns a variant containing an array. Declaring the dimension for WinWrap Basic and Hummingbird Basic is very similar, but there are differences in Microsoft Visual Basic.

Cbool—Converts a number or string to a Boolean value.

Note: Boolean is not supported by Hummingbird Basic.

Cbyte—Converts a number or string value to a byte value.

Note: Boolean is not supported by Hummingbird Basic.

Ccur—Converts a number or string value to a currency value.

Cdate—Converts a number or string value to a date value.

Note: Boolean is not supported by Hummingbird Basic.

CDbl—Converts a number or string value to a double-precision real value.

Cint—Converts a value to an integer by rounding.

CLng—Converts a value to a Long by rounding.

CSng—Converts a value to a single-precision float point.

CStr—Converts a number or string value to a string value.

Cvar—Converts a value to a variant.

CVDate—Converts a value to a variant date.

CVErr—Converts to a variant that contains an error code. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Data Type Group

Boolean—A true or false value. It uses integer values to represent true and false.

Byte—An 8-bit unsigned integer value.

Currency—Currency variables are stored as 64-bit numbers in an integer format.

Date—The whole part represents the date, while the fractional part is the time of day.

Double—A 64-bit real value.

Integer—A 16-bit integer value.

Long—A 32-bit integer value.

Object—An object reference value.

PortInt—A portable integer value.

Single—A 32-bit real value.

String—An arbitrary-length string value.

String*n—A fixed-length (n) string value. Not supported by Hummingbird Basic.

Variant—An empty, numeric, currency, date, string, object, error code, null, or array value.

DDE Group

DDEExecute—Sends one or more commands to an application using a Dynamic Data Exchange (DDE) channel.

DDEInitiate—Opens a DDE channel and returns the DDE channel number (1,2, and so on).

DDEPoke—Sends data to an application on an open DDE channel.

DDERequest—Returns data from an application on an open DDE channel.

DDETerminate—Closes the specified DDE channel.

DDETerminateAll—Terminates all open DDE channels. Not supported by Hummingbird Basic.

Declaration Group

Attributes—Sets or returns a value that indicates one or more characteristics of a Field, Relation. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Declare—Used at the module level to declare references to external procedures in a dynamic-link library (DLL).

Class_Initialize—A class module-initialization subroutine. Each time a new instance is created for a class module, the Class_Initialize subroutine is called. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Class_Terminate—A class module-termination subroutine. Each time an instance is destroyed for a class module, the Class_Terminate subroutine is called. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Const—Declares constants for use in place of literal values.

Deftype—Specifies the default data type for one or more variables.

Dim—Declares variables and allocates storage space.

Enum—Declares a type for an enumeration. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic

Friend—Modifies the definition of a procedure in a class module to make the procedure callable from modules that are outside the class, but part of the project within which the class is defined.

Functions—Declares the name, argument, and code that forms the body of a function procedure.

Global—An Application object that enables you to access application-level properties and methods.

Option Explicit—Specifies that all variables in a module must be explicitly declared.

Object_Initialize—An object module-initialization subroutine. Not supported by Hummingbird Basic.

Object_Terminate—An object module-termination subroutine. Not supported by Hummingbird Basic.

Private— Creates arrays (or simple variables) that are available to an entire macro or module, but not other macros or modules. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Public—Creates arrays (or simple variables) that are available to an entire macro or module, but not other macros or modules. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

ReDim—Changes the upper and lower bounds of a dynamic array's dimension.

Static—Used inside procedures to declare variables and allocate storage space.

Sub—Declares the name, arguments, and code that form the body of a subprocedure.

Type—Declares a user-defined type, which can then be used in the Dim statement to declare a record variable.

DefType Statement

Used at the module level to set the default data type for variables, arguments passed to procedures, and the return type for Function and Property Get procedures whose names start with the specified characters.

DefBool—Boolean is not supported by Hummingbird Basic.

DefByte—Byte is not supported by Hummingbird Basic.

DefCur—Currency.

DefDate—Date is not supported by Hummingbird Basic.

DefDb—Double.

DefInt—Integer.

DefLong—Long.

DefObj—Object is not supported by Hummingbird Basic.

DefVar—Variant.

DeleteSetting—Deletes a section or key setting from an application's entry in the Windows registry. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Dialog Methods and Statements

You can use Dialog methods and statements only when there is an active dialog box on the screen. In other words, only the method that was associated with the active dialog box in the BeginDialog statement can call these methods.

DialogFunc—Implements the dynamic dialog capabilities. Not supported by Hummingbird Basic.

DlgControlId—Returns the numeric ID of a dialog box control.

DlgCount—Returns the number of dialog box items in the dialog box. Not supported by Hummingbird Basic.

DlgEnable (method)—Indicates whether a control is enabled or disabled.

DlgEnable (statement)—Enables or disables a dialog box control.

DlgEnd—Closes the dialog box.

DlgFocus—Sets the focus to a dialog box control.

DlgListBoxArray (method)—Returns the contents of a list box or combo box.

DlgListBoxArray (statement)—Sets the contents of a list box or combo box.

DlgName—Returns the field name. Not supported by Hummingbird Basic.

DlgNumber—Returns the number of the expression that returns a string result. Not supported by Hummingbird Basic.

DlgSetPicture—Changes the picture in a picture dialog box control for the current dialog box. Not supported by Hummingbird Basic.

DlgText (method)—Returns the text associated with a dialog box control.

DlgText (statement)—Sets the text associated with a dialog box control.

DlgType—Returns a string value indicating the type of expression that returns a numeric result. Not supported by Hummingbird Basic.

DlgValue (method)—Returns the value associated with a dialog box control.

DlgValue (statement)—Sets the value associated with a dialog box control.

DlgVisible (method)—Indicates whether a control is enabled or disabled.

DlgVisible (statement)—Shows or hides a dialog box control.

Error Handling Group

Err Object—The following is not supported by Hummingbird Basic. Can be found in Microsoft Visual Basic (simulates the occurrence of an error).

Err[.Number]—The error code for the last error event.

Err.Description—The description of the last error event.

Err.Source—The error-source file name of the last error event.

Err.HelpFile—The Help file name of the last error event.

Err.HelpContext—The Help context ID of the last error event.

Err.Clear—Clears the last error event.

Err.Raise—Raises an error event.

Err.LastDLLError—For 32-bit Windows, returns the error code for the last DLL call. For 16-bit Windows, always returns 0.

Error—Returns the error message that corresponds to the specified error code.

On Error—Specifies the location of an error-handling routine within the current procedure.

Resume—Resumes execution after an error-handling routine is finished.

File Group

ChDir—Changes the default directory for the specified drive. It does not change the default drive.

ChDrive—Changes the default drive.

Close—Closes a file, concluding input/output to that file.

CurDir—Returns the path (including the drive letter) of the current default directory for the specified drive.

Dir—Returns a string representing the name of a file, directory, or folder that matches a specified pattern or file attribute or the volume label of a drive.

EOF—Returns a value indicating whether the end of a file has been reached.

FileAttr—Returns information about an open file. Depending on the attribute chosen, this information is either the file mode or the operating system handle.

FileCopy—Copies a file.

FileDateTime—Returns a string that indicates when a specified file was last modified.

FileLen—Returns a Long that indicates the length of the specified file.

FreeFile—Returns the lowest unused file number.

Get—Reads a variable from a file opened in Random or Binary mode.

GetAttr—Returns an integer representing the attributes of a file, directory, or folder.

Input (method)—Returns a string containing characters from a file opened in Input or Binary mode.

Input (statement)—Reads data from an open sequential file and assigns the data to variables.

Kill—Deletes files from a disk.

LineInput—Reads a single line from an open sequential file and assigns it to a string variable.

Loc—Returns a Long specifying the current read/write position within an open file.

Lock—Controls access by other processes to all or part of a file opened using the Open statement.

LOF—Returns a Long representing the size, in bytes, of a file opened using the Open statement.

MkDir—Makes a new directory.

Name—Renames a file.

Open—Opens a file or device for input or output.

Print—Writes display-formatted data to a sequential file.

Put—Writes a variable to a file opened in Random or Binary mode.

Reset—Closes all open disk files and writes any data still remaining in the operating-system buffer to disk.

RmDir—Removes an existing directory or folder.

Seek (method)—Returns a Long specifying the current read/write position within a file opened using the Open Statement.

Seek(statement)—Sets the position for the next read/write operation within a file opened using the Open statement.

SetAttr—Sets attribute information for a file.

UnLock—Controls access to an open file.

Write—Writes data to an open sequential file.

Flow Control Group

Call—Transfers control to a subprogram procedure.

Case—Executes one of a series of statement blocks, depending on the value of an expression.

Choose—Selects and returns a value from a list of arguments. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Do Loop—Repeats a block of statements while a condition is TRUE or until a condition becomes TRUE.

Note: Hummingbird Basic does not support Boolean values, so it interprets TRUE as non-zero and FALSE as zero.

Each—Repeats a group of statements for each element in an array or collection.

Note: The For Each Next Statement part is not supported by Hummingbird Basic.

End—An instruction that causes a macro to terminate immediately.

Exit—An instruction that causes a macro to continue without carrying out some or all of the remaining instructions.

For Next Each—Repeats a group of statements for each element in an array or collection. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

For Next—Repeats a group of statements a specified number of times.

Goto—The Goto statement sends control to a label.

If Then—Executes alternative blocks of program code based on one or more expressions.

MacroDir—Returns the directory of the current macro. A run-time error occurs if the current macro has never been saved. Not supported by Hummingbird Basic.

MacroRun—Plays a macro. Execution continues at the following statement after the macro has completed. Not supported by Hummingbird Basic.

Select Case—Executes one of a series of statement blocks, depending on the value of an expression.

Stop—Halts program execution.

While—Controls a repetitive action.

Math Group

Abs—Returns the absolute value.

Atn—Returns the angle (in radians) corresponding to the arc tangent of the specified numeric expression.

Cos—Returns the cosine of an angle.

Exp—Returns the exponential.

Fix—Returns the integer value of the number value.

Int—Returns a value of the type passed to it containing the integer portion of a number.

Log—Returns a Double specifying the natural logarithm of a number.

Randomize—Initializes the random-number generator.

Rnd—Returns a random number greater than or equal to zero and less than one.

Sgn—Returns a value indicating the sine of the number.

Sin—Returns a Double specifying the sine of an angle.

Sqr—Returns a Double specifying the square root of a number.

Tan—Returns a Double specifying the tangent of an angle.

TimeValue—Returns a variant (vate) containing the time.

Miscellaneous Group

'—Used to include explanatory remarks in a program.

AboutWinWrapBasic—Shows the WinWrap Basic About box. Not supported by Hummingbird Basic.

AppActivate—Activates an application window. Activates an application’s top-level window title window.

Beep—Produces a single, short beeping tone through the computer’s speaker.

CallersLine—Returns the line of a caller as a text string. Not supported by Hummingbird Basic.

Clipboard—Provides access to the system Clipboard.

Command—Returns a string containing the command line specified when the MAIN subprogram was invoked.

Debug—Sends output to the Immediate window at run time. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic

DoEvents—Yields execution so that the operating system can process other events.

Environ—Returns the string associated with an operating system environment variable.

IIf—Returns the value of the indicated by an expression that returns a numerical result. Not supported by Hummingbird Basic.

QBColor—Returns a Long representing the RGB color code corresponding to the specified color. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

RGB—Returns a Long whole number representing an RGB color value. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

SendKeys—Sends one or more keystrokes to an active window as if they were typed at the keyboard.

Shell—Runs an executable program. If successful, it returns a variant (Double) representing the program's task ID; otherwise, it returns zero.

Wait—Waits for delay seconds. Not supported by Hummingbird Basic.

Object Group

CreateObject—Creates a new Ole2 automation.

GetObject—Returns an Ole2 object associated with the file name or the application name.

With—Executes a series of statements on a specified variable.

Operators Group

^—Exponentiation.

-,+—Unary minus and plus. The + operator is also used for string concatenation.

***,/—**Numeric multiplication or division. For division, the result is a Double.

\—Integer division. The operand can be Integer or Long.

Mod—Modulus or Remainder. The operand can be Integer or Long.

&—String concatenation.

>,<,=,<=.>=,==—Numeric or string comparison.

Not—Unary Not.

And—Operands can be Integer or Long.

Or—Inclusive Or.

Xor—Exclusive Or.

Eqv—Equivalence.

Imp—Implication.

Is—Compares two object reference variables.

Like—Compares two strings.

Rem—Includes explanatory remarks in a program.

Settings Group

DeleteSetting—Deletes a section or key setting from an application's entry in the Windows registry.

GetAllSettings—Returns a list of key settings and their respective values from an application's entry in the Windows registry. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

GetSetting—Returns a key setting value from an application's entry in the Windows registry. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Save Setting—Saves or creates an application entry in the Windows registry. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

String Group

Asc—Returns an integer corresponding to the ANSI code of the first character in the specified string.

AscB—Returns the first byte. Not supported by Hummingbird Basic.

AscW—Returns the Unicode number.

Chr—Returns a one-char string for the ASCII value.

ChrB—Returns a single-byte ACSII string. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

ChrW—Returns a single char Unicode string. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Format—Returns a formatted string of an expression based on a given format.

Hex—Returns the hexadecimal representation of a number, as a string.

InStr—Returns a variant (Long) specifying the position of the first occurrence of one string within another.

IntStrB—Returns the byte index specifying the position of the first occurrence of one string within another. Not supported by Hummingbird Basic.

LCase—Returns a string that has been converted to lowercase.

Left—Returns a variant (string) containing a specified number of characters from the left side of a string.

LeftB—Returns bytes containing a specified number of characters from the left side of a string. Not supported by Hummingbird Basic.

Len—Returns a Long containing the number of characters in a string.

LenB—Returns a byte containing the number of characters in a string. Not supported by Hummingbird Basic.

LTrim—Returns a copy of the source string, with all leading spaces removed.

Mid (method)—Returns a variant (string) containing a specified number of characters from a string.

Mid (statement)—Replaces a specified number of characters in a variant (string) variable with characters from another string.

MidB—Returns a byte containing a specified number of characters from a string. Not supported by Hummingbird Basic.

Oct—Returns a variant (string) representing the octal value of a number.

Right—Returns a string of a specified length copied from the right-most character of the string expression.

RightB—Returns a byte of a specified length copied from the right-most character of the string expression. Not supported by Hummingbird Basic.

Rtrim—Returns a copy of the source expression with all trailing spaces removed.

Space—Returns a variant (string) consisting of the specified number of spaces.

Str—Returns a string representation of a number.

StrConv—Returns a variant (string) converted as specified. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

String—Returns a variant (string) containing a repeating character string of the length specified.

Trim—Returns the string with leading and trailing spaces removed.

Ucase—Returns a copy of a string after all lowercase letters have been converted to uppercase.

Val—Returns the numeric value of the first number found in the specified string.

TimeDate Group

Date—Returns a string representing the current date.

DateAdd—Returns a variant (date) containing a date to which a specified time interval has been added. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

DateDiff—Returns a variant (Long) specifying the number of time intervals between two specified dates. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

DatePart—Returns a variant (integer) containing the specified part of a given date. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

DateSerial—Returns a variant (date) for the specified year, month, and day.

DateValue—Returns a date value for the string specified.

Day—Returns the day-of-the-month component of a date-time value.

Hour—Returns the hour-of-day component (0–23) of a date-time value.

Minute—Returns the minute component of a date-time value.

Month—Returns the month component of a date-time value.

Now—Returns the current date and time.

Second—Returns the second component of a date-time value.

Time—Returns a string representing the current time.

Timer—Returns the number of seconds past midnight.

TimerSerial—Returns a variant (date) containing the time for a specific hour, minute and second.

WeekDay—Returns the day of the week for the specified date-time value.

Year—Returns the year component (1–12) of a date-time value.

User Dialog Group

Begin—Starts the dialog-box declaration for a user-defined dialog box.

Begin Dialog—Begins and ends a dialog-box declaration.

CancelButton—Used in the interactive dialog box.

CheckBox—Used in the interactive dialog box.

ComboBox—Used in the interactive dialog box.

Dialog (statement)—Displays a dialog box.

DialogFunc—Implements the dynamic dialog capabilities. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

DropListBox—Defines a drop-down list box item.

GroupBox—Defines and draws a box that encloses sets of dialog box items, such as option boxes and check boxes.

ListBox—Displays a list of items from which the user can select one or more.

OKButton—Defines an OK button dialog box control. See Dialog Box Definition.

OptionButton—Defines an option button item.

OptionGroup—Groups a series of option buttons under one heading in a dialog box.

Picture—Defines a picture control in a custom dialog box.

PushButton—Defines a custom push button.

Text—Places lines of text in a dialog box.

TextBox—Sometimes called an edit field or edit control. Displays information entered at design time, entered by a user, or assigned to the control in the code at run time.

User Input Group

Dialog (method)—Displays a dialog box and returns a number for the button selected.

GetFilePath—Displays a dialog box and gets a file path from the user. The returned string is a complete path and file name. Not supported by Hummingbird Basic.

InputBox—Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns a string containing the contents of the text box.

MsgBox (method)—Returns an integer value indicating which button the user selected.

MsgBox (statement)—Displays a message in a dialog box. If a message box requires buttons in addition to OK, use the MsgBox method instead.

Variable Info Group

IsArray—Returns a Boolean value indicating whether a variable is an array. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

IsDate—Determines whether a value is a legal date.

IsEmpty—Returns a value that identifies whether a variant has been initialized.

IsError—Returns a Boolean value indicating whether a variable has been initialized. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

IsMissing—Returns a Boolean value indicating whether an optional variant argument has been passed to a procedure. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

IsNull—Returns a value that identifies whether an expression has resulted in a null value.

IsNumeric—Returns a value that signifies whether a variant is of numeric type.

IsObject—Returns a Boolean value indicating whether an identifier represents an object variable. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Lbound—Returns a Long containing the smallest available subscript for the indicating dimension of an array.

TypeName—Returns a string indicating the type of value stored. Not supported by Hummingbird Basic.

Ubound—Returns the upper bound of the subscript range for the specified array.

VarType—Returns the ordinal number representing the type of data currently stored in the variant. A string representation with a prefix of vb of the value is used in WinWrap Basic but not in Hummingbird Basic. Instead, Hummingbird Basic includes the MsgBox Instruction/Method, String Data Type, Attribute definition, Shell Method, Var Type, Weekday Method, and StrConv Method. It also uses the numerical values of the string representation with prefix vb.

File Transfer Options

You can use this option to set file-transfer options. Remember that if you are using CMS, you must precede the options list with an open parenthesis.

Example SendFile "C:\CONFIG.SYS" "CONFIG SYS A1" "(ASCII CRLF"

Related Topics

General Options

CMS-specific Options on Upload

TSO-specific Options on Upload

MUSIC-specific Options on Upload

General Options

The General Options group box lets you specify whether you are transferring text or binary files or whether you want to append the file you are transferring to an existing file.

ASCII—Specifies ASCII to EBCDIC translation. Check this option when transferring files.

CRLF—This is the carriage return and line feed code. This code is necessary for viewing and editing text and source files, such as SCRIPT files. It is not required for binary files. Check this option when transferring text files.

APPEND—Specifies that you want to add the file you are sending to the end of the host file. Omit this option if you want the file to replace an existing host file.

Related Topics

CMS-specific Options on Upload

TSO-specific Options on Upload

MUSIC-specific Options on Upload

CMS-Specific Options on Upload

The following CMS-specific options let you set the record format.

RECFM x—The record format of the resulting CMS file where x = V (variable) or F (fixed). If you omit this option, the file will contain variable-length records if you specify CRLF; otherwise, it will contain fixed-length records.

LRECL n—The record length of the resulting CMS file, where *n* is the logical record length. Include a record length only if you want the resulting file to have a record length other than 80. If you omit this option, the file will have a record length of 80.

Related Topics

General Options

Xfer Options

TSO-specific Options on Upload

MUSIC-specific Options on Upload

TSO-Specific Options on Upload

The following TSO-specific options let you set transfer options:

(MEMBER)—If you are uploading the file to a partitioned data set, you can append the member name to the host file name.

/PASSWORD—If the data set contains a password, you can append it to the host file name.

RECFM(x)—The record format of the resulting TSO data set, where x = V (variable), F (fixed), or U (undefined). If you omit this option, the file will contain variable-length records if you specified CRLF; otherwise, it will contain fixed-length records. Do not use this option with the MEMBER option.

LRECL(n)—The record length of the resulting TSO data set, where n = 1 through 132. If you omit this option, the record length is set at 80. Do not use this option with the MEMBER option.

BLKSIZE(n)—The block size of the resulting TSO data set. If you omit this option, the block size will be the same as the record length. Do not use this option with the MEMBER option.

SPACE(n1,n2) units—The amount of space to be allocated for the resulting TSO data set (assuming it is a new one), where:

· n1 = primary quantity in the units specified

· n2 = increment in the units specified (if the primary space is insufficient)

· units = AVBLOCKS, TRACKS, or CYLINDERS

The units parameter is optional. These values are similar to the values in the TSO ALLOCATE command. If you omit this option, you will get the space for one block, with the length of the block being set by the BLKSIZE or LRECL options. Do not use this option with the MEMBER option.

Related Topics

General Options

Xfer Options

CMS-specific Options on Upload

MUSIC-specific Options on Upload

MUSIC-Specific Options on Upload

The following MUSIC-specific options let you set transfer options:

LRECL(n)—The record length of the resultant MUSIC save file where n = 1 through 32767. If you omit this option, the record length is set at 80.

RECFM(x)—The record format of the resultant MUSIC save file where x = V (variable), F (fixed), VC (variable compressed), or FC (fixed compressed). If you omit this option, the file will have variable length records.

SPACE(n)—The primary space allocation for the resultant MUSIC save file where n = 1 through 8000 (Kb). If you omit this option, the default primary allocation will be 40.

Related Topics

General Options

Xfer Options

CMS-specific Options on Upload

TSO-specific Options on Upload

IsXfer Host Property 3270

This read-only property returns a TRUE value if a file transfer is currently taking place.

Syntax	bVal = IsXfer
Group	Host Object
Mode	3270

Example	Sub Main
	Dim HE as Object
	Set HE = CreateObject("HostExplorer")
	
	HE.CurrentHost.SendFile "C:\CONFIG.SYS", "CONFIG SYS A1", "(ASCII CRLF"
	If HE.CurrentHost.IsXfer Then
	MsgBox "Wait until File Transfer is complete"
	End If
	End Sub

Related Topics

ReceiveFile Host Method

SendFile Host Method

WaitXfer Host Method

XferCount Host Property

XferRC Host Property

SendFile Host Method 3270 VT

You can use this method to initiate a file transfer (upload) to the host system.

In 3270 mode, the first parameter specifies the local file to be uploaded, the second parameter specifies the host filename to receive the file, and the last parameter specifies the host file transfer options. 

In VT mode, the first parameter specifies the local file to receive the data and the second parameter specifies the file transfer protocol to use.

Syntax

SendFile pcfilename$, hostfilename$, options$
SendFile pcfilename$, iXferProtocol%

pcfilename$: PC filename to upload to host

hostfilename$: Host filename to receive file

options$: optionsXFEROPTIONS for this file transfer

iXferProtocol%: File transfer protocol to use. Refer to HEBASIC.EBH for values.

Group Host Object

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

' 3270 mode upload
HE.CurrentHost.SendFile "C:\CONFIG.SYS", "CONFIG SYS A1", "( ASCII CRLF"
HE.CurrentHost.WaitXfer

' VT mode upload
HE.CurrentHost.SendFile "C:\CONFIG.SYS", ZMODEM
HE.CurrentHost.WaitXfer
End Sub
```

Related Topics

IsXfer Host Property

ReceiveFile Host Method

WaitXfer Host Method

XferCount Host Property

XferRC Host Property

WaitXfer Host Method 3270

You can use this method to wait until a file transfer is completed. This synchronizes the macro with the file transfer.

Syntax	WaitXfer
Group	Host Object
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.SendFile "C:\CONFIG.SYS", "CONFIG SYS A1", "(ASCII CRLF" HE.CurrentHost.WaitXfer End Sub

Related Topics

IsXfer Host Property

XferCount Host Property

XferRC Host Property

XferCount Host Property 3270 5250 VT

You can use this property to return the number of bytes transferred in the current or last file transfer.

Syntax	iBytes& = XferCount
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iBytes& = HE.CurrentHost.XferCount End Sub

Related Topics

IsXfer Host Property

WaitXfer Host Method

XferRC Host Property

XferRC Host Property 3270

This read-only property returns the error code for the last file transfer. The return codes are consistent with those found in the EHLLAPI and DDE application program interfaces.

Syntax	iRc% = XferRC
Group	Host Object
Mode	3270

Example	Sub Main
	Dim HE as Object
	Set HE = CreateObject("HostExplorer")
	
	HE.CurrentHost.SendFile "C:\CONFIG.SYS", "CONFIG SYS A1", "(ASCII CRLF"
	HE.CurrentHost.WaitXfer
	
	If HE.CurrentHost.XferRC <> 3 Then
	MsgBox "Some error occurred during file transfer. Rc: " + Str$(HE.CurrentHost.XferRC)
	End If
	End Sub

Related Topics

IsXfer Host Property

WaitXfer Host Method

XferCount Host Property

Connect Host Method 3270 5250 VT

You can use this method to connect a session to a host system. You can only make this call if your current session does not already have a host connection. This method maps directly to the File, Connect menu.

You can set the host system name and other connection parameters through the Cfg3270 and CfgVT objects.

Syntax	Connect
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Cfg3270.Host = "MyHost" HE.CurrentHost.Cfg3270.TCPPort = 23 HE.CurrentHost.Connect End Sub

Related Topics

Cfg3270 Objects List

CfgVT Objects List

ConnectErrorStatus Host Property

ConnectRC Host Property

Disconnect Host Property

WaitConnected Host Method

Cfg3270 Objects List 3270

The Cfg3270 object lets you configure all aspects of a 3270 session. Unlike other objects, the Cfg3270 object exists only to group 3270 configuration methods. It has no properties of its own. You can access configuration methods directly using the With statement or by creating an object for easy reference.

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

He.CurrentHost.Cfg3270.CursorType = CURSOR_BLOCK
End Sub

'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

With He.CurrentHost.Cfg3270
.CursorType = CURSOR_BLOCK
End With
End Sub

'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
Dim Cfg as Object
Set Cfg = HE.CurrentHost.Cfg3270

Cfg.CursorType = CURSOR_BLOCK
End Sub
```

CfgVT Objects List VT

The CfgVT object lets you configure all aspects of a VT session. Unlike other objects, the CfgVT object exists only to group VT configuration methods. It has no properties of its own. You can access configuration methods directly, using the With statement or by creating an object for easy reference.

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.CfgVT.CursorType = CURSOR_BLOCK
End Sub

'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

With HE.CurrentHost.CfgVT
.CursorType = CURSOR_BLOCK
End With
End Sub

'$include:"-E\hebasic.ebh"

Sub Main
Dim Cfg as Object
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
Set Cfg = HE.CurrentHost.CfgVT

Cfg.CursorType = CURSOR_BLOCK
End Sub
```

ConnectErrorStatus Host Property VT

This read-only property returns the error status for the last connect request. Valid return codes are:

· 0—No error

· 1—Busy signal

· 2—No answer

· 3—No dial tone

· 4—Unspecified problem after dialing

Syntax	iRc% = ConnectErrorStatus
Group	Host Object
Mode	3270, 5250, VT

Example	Sub Main
	Dim HE as Object
	Set HE = CreateObject("HostExplorer")
	
	HE.StartSession "Default.VT"
	
	If HE.CurrentHost.ConnectRC = 0 Then
	MsgBox "Error occurred during connect. Rc: " + Str$(HE.CurrentHost.ConnectErrorStatus)
	End If
	End Sub

Related Topics

Connect Host Property

ConnectRC Host Property

Disconnect Host Property

WaitConnected Host Method

ConnectRC Host Property 3270 5250 VT

This read-only property returns the connection status for the last connect request. Valid return codes are:

· 0—Not Connected

· 1—Connecting

· 2—Connected

Syntax	iRc% = ConnectRC
Group	Host Object
Mode	3270, 5250, VT

Example	Sub Main
	Dim HE as Object
	Set HE = CreateObject("HostExplorer")
	
	HE.StartSession "Default.3270"
	
	If HE.CurrentHost.ConnectRC <> 2 Then
	MsgBox "Error occurred during connect. Rc: " + Str$(HE.CurrentHost.ConnectRC)
	End If
	End Sub

Related Topics

Connect Host Property

ConnectErrorStatus Host Property

Disconnect Host Property

WaitConnected Host Method

Disconnect Host Method 3270 5250 VT

You can use this method to disconnect a session from a host system. You can call this method only if your session is currently connected. This method maps directly to the Disconnect item on the File menu.

Syntax	Disconnect
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Disconnect End Sub

Related Topics

Connect Host Method

WaitConnected Host Method 3270 5250 VT

You can use this method to synchronize the connection process to a host. This method will wait until the connection is complete or for up to 'iTimeout' seconds. Valid return codes are:

· 0—Connection complete.

· 1—Timer expired. Emulator is not connected.

Syntax	iRc% = WaitConnected(iTimeout&) iTimeout&: timeout for method in seconds
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.StartSession "Default.3270" iRc% = HE.CurrentHost.WaitConnected(60) End Sub

Related Topics

Connect Host Method

ConnectErrorStatus Host Property

ConnectRC Host Property

Restore Host Method 3270 5250 VT

You can use this method to restore the host object. This is a non-maximized mode.

Syntax	Restore
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Restore the current host screen HE.CurrentHost.Restore End Sub

Related Topics

Maximize Host Method

Minimize Host Method

Maximize Host Method 3270 5250 VT

You can use this method to maximize the host session window.

Syntax	Maximize
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Maximize the current host screen HE.CurrentHost.Maximize End Sub

Related Topics

Maximize Host Method

Restore Host Method

Minimize Host Method 3270 5250 VT

You can use this method to minimize the host session window.

Syntax	Minimize
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") ' Minimize the current host screen HE.CurrentHost.Minimize End Sub

Related Topics

Minimize Host Method

Restore Host Method

ConnectBy Host Property 3270 5250 VT

You can use this property to set or retrieve the connection type. The following definitions exist in the HEBASIC.EBH header:

Const IO_TELNET = 0 ' All terminal types

Const IO_MODEM =1 ' VT Only

Const IO_MSSNA = 2 ' 3270 Only

Const IO_NWSAA = 3 ' 3270 Only

Const IO_DEMOLINK = 4 ' All terminal types

Syntax iMode% = ConnectBy

Group Host Object

Mode 3270, 5250, VT

Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Dim Host as Object

Set HE = CreateObject( "HostExplorer" )
Set Host = HE.Host( HE.NewSession)

Host.TerminalMode = TERMINAL_3270 ' Set terminal mode first!
Host.Model = TN3270_MODEL_2 ' Set terminal model next
Host.ConnecBy = IO_TELNET ' Set connection method
Host.Cfg3270.Host = "MyHost" ' Set host name
Host.Connect ' OK, connect to host saved
End Sub
```

Row Host Method 3270 5250 VT

You can use this method to retrieve the contents of the screen row-by-row. The numeric parameter specifies which row to retrieve. The value for the parameter can range between 1 and the number of rows in the presentation space. Here, you can convert nulls to blanks.

Syntax

```
szRowText$ = Row( iRow% )

iRow%: row to retrieve text

szRowText$: string for row retrieved
```

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

SecondRow$ = HE.CurrentHost.Row(2)
MsgBox "The content of the specified row in the presentation space is " &
SecondRow, , "Output"
End Sub
```

Related Topics

Text Host Property

TextRC Host Property

Text Host Property 3270 5250 VT

You can use this property to retrieve and update the entire screen as a string. Nulls are converted to blanks. The size of the string returned is the number of columns * the number of bytes.

Syntax	szText$ = Text
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szTxt$ = HE.CurrentHost.Text MsgBox szTxt$, "Screen Content" End Sub

Related Topics

Row Host Method

TextRC Host Property

TextRC Host Method 3270 5250 VT

You can use this method to retrieve part of the screen as a string. You must specify a valid row and column. The first position on the screen is Row 1 Column 1. The iLength% value can be:

· 0—Copy to End of Field.

· -1—Copy to End of Line.

· -2—Copy to End of Word.

· -3—Copy to End of Screen.

· >0—Exact length specified.

Syntax

szText$ = TextRC(iRow%, iCol%, iLength%)

iRow%: row of text to retrieve

iCol%: column of text to retrieve

iLength%: length of text to retrieve

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szTxt$ = HE.CurrentHost.TextRC( 5, 5, 20 )
End Sub
```

Related Topics

Row Host Method

Text Host Property

Cursor Host Property 3270 5250 VT

You can use this property to get or set the cursor location. The location is expressed with an absolute number from 1 to the screen size. If the value is outside this range, the system ignores the call. In VT mode, the Cursor property is read-only.

Syntax	Cursor = iPos%
Group	Host Object
Mode	3270, 5250, VT (Read-Only)
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iPos = HE.CurrentHost.Cursor HE.CurrentHost.Cursor = 1760 MsgBox "The position of the cursor is at " + iPos, , "Information" End Sub

Related Topics

CursorRC Host Property

CursorRC Host Property 3270 5250 VT

You can use this property to set the cursor location using row and column values. If a value is outside the valid range, the system ignores the call. The first position on the screen is row 1 column 1.

Syntax	CursorRC iRow%, iCol% iRow%: row position iCol%: column position
Group	Host Object
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.CursorRc 5, 6 ' Set cursor to Row 5, Col 6 End Sub

Related Topics

Cursor Host Property

Device3279 Host Property 3270

This read/write property lets you set the 3270 device as FALSE (3278) or TRUE (3279). For this value to take effect, you must first disconnect and reconnect the session.

Syntax	b3279Device = Device3279
Group	Host Object
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") b3279Device = HE.CurrentHost.Device3279 MsgBox "The 3270 Device is set to " + b3270Device, , "Information" End Sub

Related Topics

EAB Host Property

Model Host Property

EAB Host Property 3270

You can use this read/write property to get or set Extended Attribute support. If the value is TRUE, the system enables Extended Attributes. Changing this value takes effect only after you disconnect and reconnect the session.

Syntax	bVal = EAB
Group	Host Object
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bEAB = HE.CurrentHost.EAB MsgBox "The Extended Attribute support has a value of " + bEAB End Sub

Related Topics

Device3279 Host Property

Model Host Property 3270 5250 VT

This read/write property returns the model type for the session or sets the model for the session. Changing this value will take effect only after you disconnect and reconnect a session.

Syntax	iModel% = Model
Group	Host Object
Mode	3270, 5250, VT
	3270—Values are: 2, 3, 4, or 5.
	5250—Values are: 2 or 5.
	TELNET—Values are:
	· TELNET_VT52
	· TELNET_VT100
	· TELNET_VT101
	· TELNET_VT102
	· TELNET_VT220
	· TELNET_VT320
	· TELNET_VT420
	· TELNET_ANSI
Example	'$Include:"-E\hebasic.ebh"
	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer")
	iModel = HE.CurrentHost.Model MsgBox "This session is of Model Type " + iModel, , "Information" End Sub

Related Topics

Device3279 Host Property

EAB Host Property

Search Host Method 3270 5250 VT

You can use this method to search the presentation space for the given search string. The bCaseSensitive flag specifies whether the search is case sensitive or not. iRow% and iCol% specify the starting position for the search. If iRow% is not passed, the cursor row is used instead. If iCol% is not passed, the cursor column is used instead. Valid return codes are:

· 0—String was found in presentation space.

· 1—String was not found in presentation space.

If you use variables in the parameter list (as opposed to constants), the method will return the row and column where the text was found.

Syntax

iRc% = Search(szSearchString$, bCaseSensitive, iRow%, iCol%)

szSearchString$: string to search for

bCaseSensitive: Case Sensitive flag – TRUE/FALSE

iRow%: Row to start search

iCol%: Column to start search

Group

Host Object

Mode

3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iRc% = HE.CurrentHost.Search( "VM/ESA", FALSE, iRow, iCol )
If iRc% = 0 Then
MsgBox "String found at Row:" & iRow & ", Col: " & iCol
End If
End Sub
```

SetFont Host Method 3270 5250 VT

You can use this method to change the session font to a specific name, width, and size. If you specify a width of 0, the system will select a visually proportioned font. The font generated may not actually match the characteristics of those you requested.

Syntax	SetFont fontname$, width%, height% fontname$: name of font width%: width of font in pixels height%: height of font in pixels
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.SetFont "Courier New",0, 12 End Sub

Related Topics

FontLarger Host Method

FontSmaller Host Method

FontLarger Host Method 3270 5250 VT

You can use this method to change the session font to the next largest font.

Syntax	FontLarger
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.FontLarger End Sub

Related Topics

FontSmaller Host Method

SetFont Host Method

FontSmaller Host Method 3270 5250 VT

You can use this method to change the session font to the next smallest font.

Syntax	FontSmaller
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.FontSmaller End Sub

Related Topics

FontLarger Host Method

SetFont Host Method

ShortName Host Property 3270 5250 VT

You can use this property to get or set the session short name. The session short name is used primarily by HLLAPI applications. Changing the shortname of a session will take effect immediately and does not require the emulator or session to be restarted.

Syntax

```
ShortName = "c"
strVal$ = ShortName
```

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

' Assign letter 'A' to this session
HE.CurrentHost.ShortName = "A"
End Sub
```

GetIOBuffer Host Method 3270

This method is used when simulating a Host connectoin through the OLE interface. When you create a connection using the Demo Link, you can retrieve the host data stream, which would normally be sent to the host. This method retrieves the host data stream field-by-field. The first field is field 1. If you pass a iFieldNum% of -1, the AID value is returned as an integer string. For example, "Enter", which is AID 0x7D, would return the string "125".

Syntax

strVal$ = GetIOBuffer(iFieldNum%)

iFieldNum%: field to query

strVal$: string returned from I/O buffer

Group Host Object

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

' Wait for 60 seconds for user to press action key
iRc% = HE.CurrentHost.WaitForIO( 60 )

If iRc% == 0 Then
cAID$ = HE.CurrentHost.GetIOBuffer( -1 )
strField1$ = HE.CurrentHost.GetIOBuffer( 1 )
End If
End Sub
```

Related Topics

LoadSlideShowFile Host Property

WaitForIO Host Method

LoadSlideShowFile Host Method 3270 5250 VT

You can use this method to load a datastream image for the current session. The file must contain the datastream in the correct binary format. You can create a datastream by using the Dlg-Save-Demo-File System Command from the emulator. Valid return codes are:

· 0—Successfully loaded datastream image.

· 1—Error loading datastream image.

Syntax	iRc% = LoadSlideShowFile(szFileName$) szFileName$: path/filename to load iRc%: return code for method
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iRc% = HE.CurrentHost.LoadSlideShowFile("C:\TEST.DEM") End Sub

Related Topics

GetIOBuffer Host Method

WaitForIO Host Method

WaitForIO Host Method 3270 5250

You can use this method (when using the Demo Link) to wait for the user to press an action key such as Enter or PFx. Valid return codes are:

· 0—Action key pressed.

· 1—Timer expired.

Syntax

```
iRc% = WaitForIO( iTimeout& )

iTimeout&: timeout for method in seconds
```

Group Host Object

Mode 3270, 5250

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

' Wait for 60 seconds for user to press action key
iRc% = HE.CurrentHost.WaitForIO( 60 )

If iRc% == 0 Then
cAID$ = HE.CurrentHost.GetIOBuffer( -1 )
strField1$ = HE.CurrentHost.GetIOBuffer( 1 )
End If
End Sub
```

Related Topics

GetIOBuffer Host Property

ShowPoppad Host Property 3270 5250 VT

You can use this property to display a poppad. Valid return codes are:

· 0—Poppad is displayed.

· 1—Poppad is empty.

· 2—Poppad is already active.

· 3—Too many poppads already active.

Syntax

iRc% = ShowPoppad(szPoppad$)

szPoppad$: name of poppad to display

Group

Host Object

Mode

3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iRc = HE.CurrentHost.ShowPoppad( "VMCMS" )
End Sub
```

ShowToolbar Host Method 3270 5250 VT

You can use this method to show the session toolbar. The toolbar cannot be shown if a mouse is not installed on the Windows system.

Syntax	ShowToolbar
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.ShowToolbar End Sub

Related Topics

HideToolbar Host Method

HideToolbar Host Method 3270 5250 VT

You can use this method to hide the session toolbar.

Syntax	HideToolbar
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.HideToolbar End Sub

Related Topics

ShowToolbar Host Method

SilentConnect Host Property 3270 5250 VT

You can use this property to get or set the Silent Connect option. This option is useful only when you construct a session manually by using the NewSession method. 

When this flag is enabled, there are no informational or error messages. These messages appear during the connection process.

Syntax SilentConnect
Group Host Object
Mode 3270, 5250, VT
Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
Dim Host as Object
Dim HE as Object
Dim Cfg as Object
Set HE = CreateObject( "HostExplorer" )

index% = HE.NewSession
Set Host = HE.Hosts(index%)
Set Cfg = Host.CfgVT ' Use CfgVT, Cfg3270, or Cfg5250
Host.TerminalMode =TERMINAL_TELNET ' Set terminal mode first!
Host.Model =TELNET_VT220 ' Set terminal model next
Host.ConnectBy = IO_TELNET ' Set transport next
Host.SilentConnect = TRUE ' Silent connection

Cfg.Host = "myhost.mydomain"
Cfg.ConnectTimeout = 30
Cfg.TCPPort = 23

Host.Connect
End Sub
```

HighlightText Host Method 3270 5250 VT

You can use this method to return text that you have highlighted on the screen by using your mouse. The parameter row% selects the row of highlighted text to return. Row 1 is the first highlighted row, row 2 is the second, and so on. Passing a row number of 0 will result in the entire highlighted block being returned with newline characters "\n" as line separators. If you specify an invalid row number, you will get a null string.

Syntax

szText$ = HighlightText(row%)

row%: highlighted row to return

szText$: string returned

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szText$ = HE.CurrentHost.HighlightText( 0 )
MsgBox "The content of the highlighted area is " & szText$, ,"Hummingbird Basic
Output"
End Sub
```

SystemColor Host Method 3270 5250 VT

You can use the SystemColor Host method to change the color composition for one of the 16 base colors used by HostExplorer. This performs the same function as accessing the Color Palette category in the Session Profile dialog box. The values for the iRed, iGreen, and iBlue parameters can vary between 0 and 255. The value for the iColor parameter can range from 0 to 15 which maps to the normal VGA base colors or you can use the following reserved names:

BLACK BLUE GREEN CYAN
RED MAGENTA BROWN WHITE
GRAY LIGHT_BLUE LIGHT_GREEN LIGHT_CYAN
LIGHT_RED LIGHT_MAGENTA YELLOW BRIGHT_WHITE

Syntax

SystemColor iColor%, iRed%, iGreen%, iBlue%

iColor%: color index to set color mix

iRed%: red color mix value

iGreen%: green color mix value

iBlue%: blue color mix value

Group Host Object

Mode 3270, 5250, VT

Example

```
'$Include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.SystemColor LIGHT_BLUE, 64, 64, 255
End Sub
```

Related Topics

HostBGColor Cfgxxxx Method

HostColor Cfgxxxx Method

HostFGColor Cfgxxxx Method

HostBGColor Cfgxxxx Method 3270 5250 VT

You can use this method to retrieve the background color for a given 3270, 5250, or Telnet system color.

Syntax	iBG% = .HostBGColor(iColor%)
Group	CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iBG% = HE.CurrentHost.Cfg3270.HostBGColor(2) MsgBox "The value for that specified background system color is " &"" & iBG% & "", "Color Information" End Sub

Related Topics

SystemColor Method

HostColor Cfgxxxx Method

HostFGColor Cfgxxxx Method

HostColor Cfgxxxx Method 3270 5250 VT

You can use this method to set the foreground and background colors for 3270, 5250, or Telnet system color.

Syntax	HostColor iColor%, iFG%, iBG%
Group	CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Cfg3270.HostColor 2, 5, 1 End Sub

Related Topics

SystemColor Method

HostBGColor Cfgxxxx Method

HostFGColor Cfgxxxx Method

HostFGColor Cfgxxxx Method 3270 5250 VT

You can use this method to retrieve the foreground color for a given 3270, 5250, or Telnet system color.

Syntax	iFG% = .HostFGColor(iColor%)
Group	CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iFG% = HE.CurrentHost.Cfg3270.HostFGColor(2) MsgBox "The value for that specified foreground system color is " &"" & iFG% & "", "Color Information" End Sub

Related Topics

SystemColor Method

HostBGColor Cfgxxxx Method

HostColor Cfgxxxx Method

TerminalMode Host Property 3270 5250 VT

This read/write property returns the terminal type that is currently defined for the session. You can set this value to change the base terminal type for an unconnected session only. Changing this value takes effect only after a session is disconnected and reconnected. You can use the following literals to test the returned value:

· TERMINAL_3270

· TERMINAL_TELNET

· TERMINAL_5250

Syntax	iMode% = TerminalMode
Group	Host Object
Mode	3270, 5250, VT
Example	'$Include:"-E\hebasic.ebh" Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iMode = HE.CurrentHost.TerminalMode MsgBox "The current Terminal Mode is " + iMode, , "Information" End Sub

Related Topics

Device3279 Host Property

EAB Host Property

Host Object Methods and Properties 3270 5250 VT

The methods and properties of the Host object let you configure and manipulate all aspects of a screen. You can access a host screen in two ways.:

· use the CurrentHost object (applies to the currently or previously active screen). This method is useful if you are writing scripts that access only one session.

· use the Hosts(n) object, where ‘n’ is a value from 1 to Hosts.Count (the total number of available sessions). You can use this method to access the nth session available.

Note: You can access host methods directly using the “With” statement, or by creating an object for easy reference.

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.InsertMode = FALSE
End Sub

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

With HE.CurrentHost
.InsertMode = FALSE
End With
End Sub

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

Dim Host as Object
Set Host = HE.Hosts(1) ' Access 1'st session

Host.InsertMode = FALSE
End Sub
```

Index Host Property 3270 5250 VT

You can use this property to return the index for the current session. This is the session control block index used internally. This corresponds to the value used in the Hosts(n) property. The first session has an index value of 0.

Syntax	Index
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") index% = HE.CurrentHost.Index MsgBox "The index of the current session is " & index%, ,"Information" End Sub

Related Topics

ShortName Host Property

InsertMode Host Property 3270 5250

This property is used to get or set insert mode for the host session.

Syntax	InsertMode = TRUE/FALSE
Group	Host Object
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bMode = HE.CurrentHost.InsertMode HE.CurrentHost.InsertMode = TRUE End Sub

TN3270 Host Property 3270 5250

This read-only property returns TRUE if HostExplorer is currently in 3270 or 5250 mode. It returns FALSE if HostExplorer is in VT or NVT terminal mode.

Syntax bVal = TN3270

Group Host Object

Mode 3270, 5250

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bMode = HE.CurrentHost.TN3270
If bMode = TRUE then
MsgBox "The current session is in 3270 or 5250 Mode."
Else
MsgBox "The current session is in VT Mode."
End If
End Sub
```

IsConnected Host Property 3270 5250 VT

This read-only property returns the value TRUE if the session is connected to a host system. It returns the value FALSE if the session is not in use.

Syntax	bVal = IsConnected
Group	Host Object
Mode	3270, 5250, VT

Example	Sub Main
	Dim HE as Object
	Set HE = CreateObject("HostExplorer")
	
	If HE.CurrentHost.IsConnected = FALSE Then
	MsgBox "Not connected to host system!", , "Information"
	Else
	MsgBox "Connected to host system!", , "Information"
	End If
	End Sub

Related Topics

Connect Host Method

ConnectErrorStatus Host Property

ConnectRC Host Property

WaitConnected Host Method

TrackMenu Host Method 3270 5250 VT

You can use this method to display the track menu at the current mouse pointer location.

Syntax TrackMenu

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.TrackMenu
End Sub
```

Update Host Method 3270 5250 VT

You can use this method to force a repaint of the session window.

Syntax	Update
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Update End Sub

Keyboard Host Property 3270 5250 VT

You can use this property to get or set the locked keyboard mode. When set to TRUE, the host keyboard locks, preventing user input. You can unlock the host keyboard by setting this value to FALSE.

Syntax	Keyboard = TRUE/FALSE
Group	Host Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iKeybLocked = HE.CurrentHost.Keyboard HE.CurrentHost.Keyboard = FALSE End Sub

WaitForString Host Method 3270 5250 VT

You can use this method to wait for a string to appear on the screen. This function is provided for compatibility purposes and has been superseded by the WaitForStringRC method 

Valid return codes are:

· 0—The string was not found.

· > 0—The row where the string was found.

Syntax

iRc% = WaitForString (szString$, iRow%, iTimeout&, bCaseSensitive)

szString$: string to search

iRow%: >0 - row to search for string
0 - search entire screen
-1 - search the row containing the cursor

iTimeout&: timeout for method in seconds

bCaseSensitive: TRUE/FALSE if search is case sensitive

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.StartSession "Solaris.VT"
HE.CurrentHost.WaitConnected 120

iRc% = HE.CurrentHost.WaitForString( "login", -1, 1, FALSE )

If iRc% > 0 Then
HE.CurrentHost.Keys "myuserid\r"

iRc% = HE.CurrentHost.WaitForString( "password", -1, 1, FALSE )

If iRc% > Then
HE.CurrentHost.Keys "mypassword\r"
MsgBox "Login successful!"
End If
End If
End Sub
```

Related Topics

WaitForStringRC Host Method

WaitForStringRC Host Method 3270 5250 VT

You can use this method to wait for a string to appear on the screen. For example, this method is useful when logging in or writing macros for UNIX systems running in Linemode. Valid return codes are:

· 0—The string was not found.

· >0—The row where the string was found.

Syntax

iRc% = WaitForStringRC(szString$, iRow%, iCol%, iFlag%, iTimeout&, bCaseSensitive)

szString$: string to search

iRow%: >0 - row to start search
0 - search entire screen
-1 - search the row containing the cursor

iCol%: column to start search

iFlag%: 0 - Search using iRow% value
iFlag%: 1 - Search only at iRow% / iCol%
(Requires iRow% >= 1)
iFlag%: 2 - Search beginning at iRow% / iCol%
(Requires iRow% >= 1)

iTimeout&: timeout for method in seconds

bCaseSensitive: TRUE/FALSE if search is case sensitive

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.StartSession "Solaris.VT"
HE.CurrentHost.WaitConnected 120

iRc% = HE.CurrentHost.WaitForStringRC( "login", -1, 1, 1, 0, FALSE )

If iRc% > 0 Then
HE.CurrentHost.Keys "myuserid\r"
iRc% = HE.CurrentHost.WaitForString( "password", -1, 1, 1, 0, FALSE )

If iRc% > Then
HE.CurrentHost.Keys "mypassword\r"
MsgBox "Login successful!"
End If
End If
End Sub
```

Related Topics

WaitForString Host Method

WaitIdle Host Method 3270 5250 VT

You can use this method to wait until the screen is idle for 'IdleTime' milliseconds. An idle screen is one that does not change in any fashion for the elapsed time. Valid return codes are:

· 0—Idletime has elapsed.

· 1—Session is closed during the waiting period.

Syntax iRc% = WaitIdle(iIdleTime&)

iIdleTime&: timeout for method in milliseconds

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.RunCmd( "PF1" )
WaitTime = HE.CurrentHost.WaitIdle( 1500 )
MsgBox WaitTime, , "Output"
End Sub
```

Related Topics

WaitForString Host Method

WaitForStringRC Host Method

WaitPSUpdated Host Method 3270 5250 VT

You can use this method to synchronize events with the Host system. This method will wait until the PS is updated, for up to 'iTimeout' seconds. Valid return codes are:

· 0—PS was updated within the timeout period.

· 1—Timer expired. PS was not updated.

Syntax

```
iRc% = WaitPSUpdated( iTimeout& )

iTimeout&: timeout for method in seconds
```

Group Host Object

Mode 3270, 5270, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

' Press Enter key
HE.CurrentHost.Keys "@E"

' Wait for host to update screen
iRc% = He.CurrentHost.WaitPSUpdated( 30 )

If iRc% > 0 Then
MsgBox "PS updated within interval specified"
End If
End Sub
```

Related Topics

PSUpdated Host Property

MouseToCursor Host Property 3270 5250 VT

You can use this R/O property to return the position of the mouse in screen coordinates. If the mouse is not in the screen area, the value returned is -1. Otherwise, the value returned is between 1 and the screen size. The screen size is the number of rows multiplied by the number of columns.

Syntax iPos% = MouseToCursor

Group Host Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iPos = HE.CurrentHost.MouseToCursor
End Sub
```

Methods and Properties of the Hosts Object 3270 5250 VT

The Hosts object consists of a collection of hosts. You can use the object to cycle through the sessions and perform actions. The following Hosts Object methods and properties are available:

Note: These methods and properties are valid for all terminal types (that is, TN3270, TN5250, and TNVT).

CloseAll Method

Count Property

Item Property

Next Property

Open Method

CloseAll Method

You can use this method to immediately terminate and close all active sessions. Make this method the last call to HostExplorer because it automatically unloads the emulator.

Syntax	Hosts.CloseAll
Group	Hosts Object
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") … HE.Hosts.CloseAll End Sub

Count Property

This property returns the total number of sessions available.

Syntax count% = Hosts.Count

Group Hosts Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

…
count% = HE.Hosts.Count
End Sub
```

Item Property

This property returns the pointer to the session (n) where "n" is a unique session ID.

Syntax

Set Host = Hosts.Item (ID%)

ID%: ID of the session

Group

Hosts Object

Mode

3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Dim Host as Object
Set HE = CreateObject( "HostExplorer" )

…
SetHost = HE.Hosts(n)
End Sub
```

Next Property

This property returns the next session after the specified session. If you specify "Null", it returns the first session.

Syntax

Set Host = Hosts.Next (HostX)

HostX: the specified host

Group

Hosts Object

Mode

3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Dim HostX as Object
Dim Host as Object
Set HE = CreateObject( "HostExplorer" )
Set HostX = HE.CurrentHost

…
Set Host = HE.Hosts.Next(HostX)
End Sub
```

Open Method

You can use the Open method to start a new session from a profile saved on disk or to create a new session from scratch without using a previously saved profile. This method takes one parameter that is the name of the profile. You can specify the ProfileName in one of two ways:

· Specify a complete directory path to the MyProfile.HEP file in the Profile\3270 directory where the user files are stored on your machine. For the appropriate directory path for your platform, refer to the list of the default locations for the user files. 

· Specify the older format if the profile is in a first level folder "MyProfile.MyFolder".

Syntax

Set Host = Hosts.Open(szProfileName$)

Set Host = Hosts.Open (TERMINAL_TELNET)

szProfileName$: name of profile to load and connect

Group Hosts Object

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Dim Host as Object
Set HE = CreateObject( "HostExplorer" )

Set Host = HE.Hosts.Open( szProfileName$ )

End Sub
```

Methods and Properties of the Cfg3270, Cfg5250, and CfgVT Objects 3270 5250 VT

The Cfg3270, Cfg5250, and CfgVT objects let you configure all aspects of their respective sessions. These objects have no properties of their own. These objects let you access configuration methods directly, using the With statement or by creating an object for easy reference.

The following object lists let you configure all aspects of a 3270, 5250, or VT session;

· Cfg3270 Objects List

· Cfg5250 Objects List

· CfgVT

The following methods and properties are available:

ActionOnExist CfgVT Property
ALA Cfg3270 Property
ALADisplayMode Cfg3270 Property
ALAInputMode Cfg3270 Property
ALAKeyboardProfileName Cfg Method
AlwaysAutoSkip Cfg3270 Property
Answerback CfgVT Property
AreaCode CfgVT Property
AutoClearXferMonitor Cfg3270 Property
AutoCopySelectedText Cfgxxxx Property
AutoMacro Cfgxxxx Property
AutoUnlockKeyboard Cfg3270 Property
AutoWrap CfgVT Property
BellMargin Cfgxxxx Property
BitMode CfgVT Property
BlinkToItalic Cfgxxxx Property
BSIsDel CfgVT Property
ClearAllTabStops Cfgxxxx Method
ClearScreenOnSizeChange CfgVT Property
ColorDisplay Cfg5250 Property
CompressBlankLinesIn Scrollback CfgVT Property
ConcealAnswerback CfgVT Property
ConnectTimeout Cfgxxxx Property
ConvertNulls Cfg3270 Property
Country CfgVT Property
CRToCRLF CfgVT Property
CursorKeyMode CfgVT Property
CursorMode Cfgxxxx Property
CursorSelectMode Cfgxxxx Property
CursorType Cfgxxxx Property
DefaultRecvDir CfgVT Property
DefaultHeight CfgVT Property
DefaultWidth CfgVT Property
Dev7171Terminal Cfg3270 Property
DirectToModem CfgVT Property
Display3DBorder Cfgxxxx Property
DisplayAttr Cfg3270 Property
DisplayControlCodes CfgVT Property
MultiLineInsert Cfgxxxx Property
Notify Cfgxxxx Property
NRCSet CfgVT Property
OnDisconnect Cfgxxxx Property
Online CfgVT Property
OptimizedDisplayMode CfgVT Property
Password Cfgxxxx Method
PrintBorder Cfgxxxx Property
PrintDocumentName Cfgxxxx Property
PrinterDeInit Cfg3270 Property
PrinterInit Cfg3270 Property
PrintFooter Cfgxxxx Property
PrintHeader Cfgxxxx Property
PrintLocation Cfgxxxx Property
PrintOIA Cfgxxxx Property
Profile Cfgxxxx Property
ProportionalFonts Cfgxxxx Property
RawAddFormFeed Cfgxxxx Property
RawCaptureMode CfgVT Property
ReplyOEM Cfg3270 Property
ReRunAutoMacro Cfgxxxx Property
RespectNumeric Cfg3270 Property
RightMargin Cfgxxxx Property
SaveAppend Cfgxxxx Property
SaveAttrsInScrollback CfgVT Property
SaveConfirm Cfgxxxx Property
SaveFile Cfgxxxx Property
SaveFontOnExit Cfgxxxx Property
SaveMode Cfgxxxx Property
SaveProfile Cfgxxxx Method
SaveProfileOnClose Cfgxxxx Property
ShowDialUpDlg CfgVT Property
ShowHotspots Cfgxxxx Property
ShowNulls Cfgxxxx Property
ShowRecvDialog CfgVT Property
SmoothScrolling CfgVT Property
SmoothScrollSpeed CfgVT Property
Sound Cfgxxxx Property

Cfg5250 Objects List 5250

The Cfg5250 object lets you configure all aspects of a 5250 session. Unlike other objects, the Cfg5250 object only exists to group 5250 configuration methods. It has no properties of its own.

You can access configuration methods directly, using the With statement or by creating an object for easy reference.

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
HE.CurrentHost.Cfg5250.CursorType = CURSOR_BLOCK
End Sub

'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

With HE.CurrentHost.Cfg5250
.CursorType = CURSOR_BLOCK
End With
End Sub

'$include:"-E\hebasic.ebh"

Sub Main
Dim Cfg as Object
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
Set Cfg = HE.CurrentHost.Cfg5250

Cfg.CursorType = CURSOR_BLOCK
End Sub
```

ActionOnExist CfgVT Property VT

You can use this property to get or set the Action On Exist setting. Possible values are:

· XFER_OVERWRITE

· XFER_RENAME

· XFER_SKIP

Syntax	iAction% = ActionOnExist%
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue% = HE.CurrentHost.CfgVT.ActionOnExist HE.CurrentHost.CfgVT.ActionOnExist = XFER_OVERWRITE End Sub

Related Topics

DefaultRecvDir CfgVT Property

FileXferProtocol CfgVT Property

ShowRecvDialog CfgVT Property

DefaultRecvDir CfgVT Property VT

You can use this property to get or set the default receive path for file downloads.

Syntax	szPath$ = DefaultRecvDir
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szPath = HE.CurrentHost.CfgVT.DefaultRecvDir HE.CurrentHost.CfgVT.DefaultRecvDir = "C:\TEMP" End Sub

Related Topics

ActionOnExist CfgVT Property

FileXferProtocol CfgVT Property

ShowRecvDialog CfgVT Property

FileXferProtocol CfgVT Property VT

You can use this property to get or set the default file transfer protocol used in VT mode. Possible values are:

· XFER_TYPE_XMODEM

· XFER_TYPE_YMODEM

· XFER_TYPE_ZMODEM

· XFER_TYPE_KERMIT

Syntax	FileXferProtocol = iValue%
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.CfgVT.FileXferProtocol HE.CurrentHost.CfgVT.FileXferProtocol = XFER_TYPE_ZMODEM End Sub

Related Topics

ActionOnExist CfgVT Property

DefaultRecvDir CfgVT Property

ShowRecvDialog CfgVT Property

ShowRecvDialog CfgVT Property VT

You can use this property to get or set the Show Receive Dialog flag.

Syntax	ShowRecvDialog = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.CfgVT.ShowRecvDialog HE.CurrentHost.CfgVT.ShowRecvDialog = TRUE End Sub

Related Topics

ActionOnExist CfgVT Property

DefaultRecvDir CfgVT Property

FileXferProtocol CfgVT Property

MultiLineInsert Cfgxxxx Property 3270 5250

You can use this property to get or set the Multiline Insert Mode option.

Syntax	MultiLineInsert = TRUE/FALSE
Group	Cfg3270, Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg3270.MultiLineInsert HE.CurrentHost.Cfg3270.MultiLineInsert = TRUE End Sub

Related Topics

MultiLineDelete Cfgxxxx Property

MultiLineDelete Cfgxxxx Property 3270 5250

You can use this property to get or set the Multiline Delete Mode option.

Syntax	MultiLineDelete = TRUE/FALSE
Group	Cfg3270, Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg3270.MultiLineDelete HE.CurrentHost.Cfg3270.MultiLineDelete = TRUE End Sub

Related Topics

MultiLineInsert Cfgxxxx Property

ALA Cfg3270 Property 3270

This property is used to enable or disable ALA support.

Syntax	ALA = TRUE/FALSE
Group	Cfg3270
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Cfg3270.ALA = TRUE bValue = HE.CurrentHost.Cfg3270.ALA End Sub

Related Topics

ALADisplayMode Cfg3270 Property

ALAInputMode Cfg3270 Property

ALAKeyboardProfileName Cfg Method

ALADisplayMode Cfg3270 Property 3270

You can use this property to get or set the ALA Display Mode used to display host ALA combinations on the screen. The setting can be either ALA_DISPLAY_SUPERIMPOSED or ALA_DISPLAY_SIDEBYSIDE.

Note: Always use the literal for assigning or comparing this value. Do not assume that the value of the literal will never change between program releases.

Syntax	ALADisplayMode = iNewMode%
Group	Cfg3270
Mode	3270
Example	'$include:"-E\hebasic.ebh" Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.Cfg3270.ALADisplayMode HE.CurrentHost.Cfg3270.ALADisplayMode = ALA_DISPLAY_SUPERIMPOSED End Sub

Related Topics

ALA Cfg3270 Property

ALAInputMode Cfg3270 Property

ALAKeyboardProfileName Cfg Method

ALAInputMode Cfg3270 Property 3270

You can use this property to get or set the ALA Input Mode used to display ALA combinations on the screen. The value can be either ALA_INPUT_SUPERIMPOSED or ALA_INPUT_SIDEBYSIDE.

Always use the literal for assigning or comparing this value. Do not assume that the value of the literal will never change between program releases.

Syntax ALAInputMode = iNewMode%

Group Cfg3270

Mode 3270

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.ALAInputMode
HE.CurrentHost.Cfg3270.ALAInputMode = ALA_INPUT_SUPERIMPOSED
End Sub
```

Related Topics

ALA Cfg3270 Property

ALADisplayMode Cfg3270 Property

ALAKeyboardProfileName Cfg Method

ALAKeyboardProfileName Cfg Method 3270

You can use this method to change the current ALA keyboard profile. This method loads the specified keyboard profile as the active session ALA keyboard profile.

Syntax ALAKeyboardProfileName szKeyProfile$

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.ALAKeyboardProfileName = "MYKEYBOARD"
End Sub
```

Related Topics

ALA Cfg3270 Property

ALADisplayMode Cfg3270 Property

ALAInputMode Cfg3270 Property

Notify Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Notify (Update Alarm) option.

Syntax Notify = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = He.CurrentHost.Cfg3270.Notify
HE.CurrentHost.Cfg3270.Notify = TRUE
End Sub
```

NRCSet CfgVT Property VT

You can use this property to get or set the National Replacement Character (NRC) set. The following string values can be set:

· None

· ISO United Kingdom

· DEC Finnish

· ISO French

· DEC French Canadian

· ISO German

· ISO Italian

· ISO Norwegian/Danish

· DEC Norwegian/Danish

· DEC Portuguese

· ISO Spanish

· DEC Swedish

· DEC Swiss

Syntax NRCSet = iNewValue$

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue = HE.CurrentHost.CfgVT.NRCSet
HE.CurrentHost.CfgVT.NRCSet = "None"
End Sub
```

OnDisconnect Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Upon Disconnect option. You can use the following literals to read or change the value:

· ONDISC_CLOSEWINDOW

· ONDISC_KEEPWINDOWOPEN

· ONDISC_RESTARTSESSION

· ONDISC_SHOWNEWSESSION

Syntax OnDisconnect = iNewDisconnectMode%

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.OnDisconnect
HE.CurrentHost.Cfg3270.OnDisconnect= ONDISC_CLOSEWINDOW
End Sub
```

Online CfgVT Property VT

You can use this property to get or set the Online option. When set to TRUE, the terminal is online and sends data to the remote system. When reset to FALSE, the terminal is offline and does not send any data to the remote system.

Syntax Online = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.Online
HE.CurrentHost.CfgVT.Online = TRUE
End Sub
```

AlwaysAutoSkip Cfg3270 Property 3270

You can use this property to get or set the Always Auto Skip option.

Syntax AlwaysAutoSkip = TRUE/FALSE

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.AlwaysAutoSkip
HE.CurrentHost.Cfg3270.AlwaysAutoSkip = TRUE
End Sub
```

OptimizedDisplayMode CfgVT Property VT

You can use this property to get or set the Display Mode option. When set (TRUE), HostExplorer operates in Optimized Display Mode. In this mode, the system updates the screen only at the end of telnet buffers received. When reset (FALSE), HostExplorer operates in Realistic Display Mode. In this mode, as HostExplorer receives data, it is displayed. Due to the overhead, Realistic Display Mode is much slower than Optimized mode.

Syntax OptimizedDisplayMode = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.OptimizedDisplayMode
HE.CurrentHost.CfgVT.OptimizedDisplayMode = TRUE
End Sub
```

Answerback CfgVT Property VT

You can use this property to get or set the Answerback string.

Syntax	Answerback = szNewAnswerbackMsg$
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szValue$ = HE.CurrentHost.CfgVT.Answerback HE.CurrentHost.CfgVT.Answerback = "Hello World" End Sub

Related Topics

ConcealAnswerback CfgVT Property

ConcealAnswerback CfgVT Property VT

You can use this property to get or set the Conceal Answerback option. If you set this property to FALSE and it was previously set to TRUE, the Answerback message will be automatically cleared to nulls.

Syntax	ConcealAnswerback = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.CfgVT.ConcealAnswerback HE.CurrentHost.CfgVT.ConcealAnswerback = TRUE End Sub

Related Topics

Answerback CfgVT Property

Password Cfgxxxx Method 3270 5250 VT

You can use this method to set the contents of the Password variable used in the macro system.

Syntax Password = szNewPassword$

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.Cfg3270.Password = "givemeaccess"
End Sub
```

AreaCode CfgVT Property VT

You can use this property to get or set the Area Code when using a Modem connection.

Syntax	AreaCode = iNewAreaCode%
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iAreaCode% = HE.CurrentHost.CfgVT.AreaCode HE.CurrentHost.CfgVT.AreaCode = 514 End Sub

Related Topics

Country CfgVT Property

DirectToModem CfgVT Property

Modem CfgVT Property

ShowDialUpDlg CfgVT Property

UseDialProperties CfgVT Property

Country Cfg3270 Property VT

You can use this property to get or set the country from which you are dialing.

Syntax	Country = szCountry$
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szCountry$ = HE.CurrentHost.CfgVT.Country HE.CurrentHost.CfgVT.Country = "Canada" End Sub

Related Topics

AreaCode CfgVT Property

DirectToModem CfgVT Property

Modem CfgVT Property

ShowDialUpDlg CfgVT Property

UseDialProperties CfgVT Property

DirectToModem CfgVT Property VT

You can use this property to get or set the DirectToModem flag. When set, this flag causes the emulator to connect to the modem immediately and bypass the dialing stage. This allows you to use a modem link as if it were a COM port.

Syntax	DirectToModem = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.CfgVT.DirectToModem HE.CurrentHost.CfgVT. DirectToModem = TRUE End Sub

Related Topics

AreaCode CfgVT Property

Country CfgVT Property

Modem CfgVT Property

ShowDialUpDlg CfgVT Property

UseDialProperties CfgVT Property

Modem CfgVT Property VT

You can use this property to get or set the modem used for the current connection. You can supply the full string as located in the Modem applet in the Control Panel or any unique substring of the modem string.

Syntax	Modem = szNewModem$
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szCurrentModem$ = HE.CurrentHost.CfgVT.Modem HE.CurrentHost.CfgVT.Modem = "MyModem" End Sub

Related Topics

AreaCode CfgVT Property

Country CfgVT Property

DirectToModem CfgVT Property

Modem CfgVT Property

ShowDialUpDlg CfgVT Property

UseDialProperties CfgVT Property

ShowDialupDlg CfgVT Property VT

You can use this property to get or set the Always Show Connect Dialog flag. This property has no effect for connections that are started from OLE Automation. This property is only used to get/set values so that you can load/save profiles.

Syntax	ShowDialupDlg = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.CfgVT.ShowDialupDlg HE.CurrentHost.CfgVT.ShowDialupDlg = TRUE End Sub

Related Topics

AreaCode CfgVT Property

Country CfgVT Property

DirectToModem CfgVT Property

Modem CfgVT Property

UseDialProperties CfgVT Property

UseDialProperties CfgVT Property VT

You can use this property to get or set the Use Area Code and Country Code flag.

Syntax	UseDialProperties = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.UseDialProperties HE.CurrentHost.CfgVT.UseDialProperties = FALSE End Sub

Related Topics

AreaCode CfgVT Property

Country CfgVT Property

DirectToModem CfgVT Property

Modem CfgVT Property

ShowDialUpDlg CfgVT Property

PrintBorder Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Print Border setting.

Syntax PrintBorder = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.PrintBorder
HE.CurrentHost.Cfg3270.PrintBorder = TRUE
End Sub
```

AutoClearXferMonitor Cfg3270 Property 3270

You can use this property to get or set the Auto Clear File Transfer Monitor option.

Syntax AutoClearXferMonitor = TRUE/FALSE

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.AutoClearXferMonitor
HE.CurrentHost.Cfg3270.AutoClearXferMonitor = TRUE
End Sub
```

Related Topics

XferBlockSize Cfg3270 Property

XferHostSystem Cfg3270 Property

XferProgramName Cfg3270 Property

XferStartAction Cfg3270 Property

XferBlockSize Cfg3270 Property 3270

You can use this property to get or set the File Transfer Block Size option.

Syntax	XferBlockSize = iNewBlockSize%
Group	Cfg3270
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.Cfg3270.XferBlockSize HE.CurrentHost.Cfg3270.XferBlockSize = 8000 End Sub

Related Topics

AutoClearXferMonitor Cfg3270 Property

XferHostSystem Cfg3270 Property

XferProgramName Cfg3270 Property

XferStartAction Cfg3270 Property

XferHostSystem Cfg3270 Property 3270

You can use this property to get or set the 3270 Host System option. The string value can be CMS, TSO, or CICS.

Syntax XferHostSystem = szNewSystem$

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue = HE.CurrentHost.Cfg3270.XferHostSystem
HE.CurrentHost.Cfg3270.XferHostSystem = "CMS"
End Sub
```

Related Topics

AutoClearXferMonitor Cfg3270 Property

XferBlockSize Cfg3270 Property

XferProgramName Cfg3270 Property

XferStartAction Cfg3270 Property

XferProgramName Cfg3270 Property 3270

You can use this property to get or set the File Transfer Program Name option.

Syntax	XferProgramName = szNewXferName$
Group	Cfg3270
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szValue = HE.CurrentHost.Cfg3270.XferProgramName HE.CurrentHost.Cfg3270.XferProgramName = "IND$FILE" End Sub

Related Topics

AutoClearXferMonitor Cfg3270 Property

XferBlockSize Cfg3270 Property

XferHostSystem Cfg3270 Property

XferStartAction Cfg3270 Property

XferStartAction Cfg3270 Property 3270

You can use this property to get or set the File Transfer Initial Action option. The possible values are:

· XFER_NOACTION

· XFER_HOME

· XFER_ENTER

· XFER_CLEAR

Syntax	XferStartAction = iNewAction%
Group	Cfg3270
Mode	3270
Example	'$include:"-E\hebasic.ebh" Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.Cfg3270.XferStartAction HE.CurrentHost.Cfg3270.XferStartAction = XFER_NOACTION End Sub

Related Topics

AutoClearXferMonitor Cfg3270 Property

XferBlockSize Cfg3270 Property

XferHostSystem Cfg3270 Property

XferProgramName Cfg3270 Property

PrintDocumentName Cfgxxxx Property 3270 5250 VT

You can use this property to set or retrieve the document name. You can insert the document name in the print header or print footer using the &F variable.

Syntax PrintDocumentName = szNewDocumentName$

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue$ = HE.CurrentHost.Cfg3270.PrintDocumentName
HE.CurrentHost.Cfg3270.PrintDocumentName = "Test Document"
End Sub
```

Related Topics

PrintFooter Cfgxxxx Property

PrintHeader Cfgxxxx Property

PrintFooter Cfgxxxx Property 3270 5250 VT

You can use this property to set or retrieve the Print Screen footer string. This string can contain the following special variables:

&C—Insert computer name.

&D—Insert current date.

&F—Insert document name string.

&P—Insert page number.

&T—Insert current line.

&U—Insert user name.

Syntax	PrintFooter = szNewPrintFooter$
Mode	3270, 5270, VT
Example	Sub Main Dim He as Object Set He = CreateObject("HostExplorer") szValue$ = He.CurrentHost.Cfg3270.PrintFooter = "&U - &D" End Sub

Related Topics

PrintDocumentName Cfgxxxx Property

PrintHeader Cfgxxxx Property

PrintHeader Cfgxxxx Property 3270 5250 VT

You can use this property to set or retieve the Print Screen header string. This string can contain the following special variables:

&C—Insert computer name.

&D—Insert current date.

&F—Insert document name string.

&P—Insert page number.

&T—Insert current line.

&U—Insert user name.

Syntax	PrintHeader = szNewPrintHeader$
Mode	3270, 5270, VT
Example	Sub Main Dim He as Object Set He = CreateObject("HostExplorer") szValue = He.Current.Cfg3270.PrintHeader ="&U - &D" End Sub

Related Topics

PrintDocumentName Cfgxxxx Property

PrintFooter Cfgxxxx Property

AutoCopySelectedText Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Auto Copy Selected Text option.

Syntax AutoCopySelectedText = TRUE/FALSE

Group Cfg3270 , Cfg5250 , CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.AutoCopySelectedText
HE.CurrentHost.Cfg3270.AutoCopySelectedText = TRUE
End Sub
```

PrinterDeInit Cfg3270 Property 3270

You can use this property to get or set the Printer Deinitialization String.

Syntax	PrinterDeInit = szNewDeInit$
Group	Cfg3270
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szValue = HE.CurrentHost.Cfg3270.PrinterDeInit HE.CurrentHost.Cfg3270.PrinterDeInit = "\eE" End Sub

Related Topics

PrinterInit Cfg3270 Property

PrinterInit Cfg3270 Property 3270

You can use this property to get or set the Printer Initialization String.

Syntax	PrinterInit = szNewInit$
Group	Cfg3270
Mode	3270
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szValue = HE.CurrentHost.Cfg3270.PrinterInit HE.CurrentHost.Cfg3270.PrinterInit = "\eE" End Sub

Related Topics

PrinterDeInit Cfg3270 Property

AutoMacro Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Auto Quick-Key/Macro Name option. The value can be the name of a Quick-Key or the name of a macro file (*.EBS or *.EBX).

Syntax AutoMacro = autoMacroName$

Group Cfg3270 , Cfg5250 , CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szAutoMacro$ = HE.CurrentHost.Cfg3270.AutoMacro
HE.CurrentHost.Cfg3270.AutoMacro = "RunMeFirst"
End Sub
```

AutoUnlockKeyboard Cfg3270 Property 3270

You can use this property to get or set the Automatic Keyboard Unlock option.

Syntax AutoUnlockKeyboard = TRUE/FALSE

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.AutoUnlockKeyboard
HE.CurrentHost.Cfg3270.AutoUnlockKeyboard = TRUE
End Sub
```

AutoWrap CfgVT Property VT

You can use this property to get or set the Auto Wrap option. When set (TRUE), HostExplorer automatically wraps lines that extend past the last column on the screen. When reset (FALSE), HostExplorer discards data that extends past the end of the screen.

Syntax AutoWrap = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.AutoWrap
HE.CurrentHost.CfgVT.AutoWrap = TRUE
End Sub
```

BellMargin Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Bell Margin column. The column value must be between 1 and the total number of columns.

Syntax	BellMargin = iNewColumn%
Group	Cfg3270 , Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue% = HE.CurrentHost.Cfg3270.BellMargin HE.CurrentHost.Cfg3270.BellMargin = 72 End Sub

Related Topics

LeftMargin Cfgxxxx Property

RightMargin Cfgxxxx Property

LeftMargin Cfgxxxx Property 3270 5250

You can use this property to get or set the Left Margin column. The column value must be between 1 and the number of columns.

Syntax	LeftMargin = iColumn%
Group	Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.Cfg3270.LeftMargin HE.CurrentHost.Cfg3270.LeftMargin = 5 End Sub

Related Topics

BellMargin Cfgxxxx Property

RightMargin Cfgxxxx Property

RightMargin Cfgxxxx Property 3270 5250

You can use this property to get or set the Right Margin column. The column value must be between 2 and the total number of columns.

Syntax	RightMargin = iNewColumn%
Group	Cfg3270, Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.Cfg3270.RightMargin HE.CurrentHost.Cfg3270.RightMargin = 72 End Sub

Related Topics

BellMargin Cfgxxxx Property

LeftMargin Cfgxxxx Property

PrintLocation Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Print Location setting. You can use the following literals to test the values returned:

· PRINT_CENTERED

· PRINT_UPPERLEFT

Syntax	PrintLocation
Group	Cfg3270, Cfg5250, CfgVT
Mode	3270, 5250, VT
Example	'$include:"-E\hebasic.ebh" Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.Cfg3270.PrintLocation HE.CurrentHost.Cfg3270.PrintLocation = Print_Centered End Sub

BitMode CfgVT Property VT

You can use this property to get or set the 8-Bit Transmission Mode option.

Syntax BitMode = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.BitMode
HE.CurrentHost.CfgVT.BitMode = TRUE
End Sub
```

PrintOIA Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Print OIA (Operator Information Area) setting.

Syntax PrintOIA = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.PrintOIA
HE.CurrentHost.Cfg3270.PrintOIA = TRUE
End Sub
```

BlinkToItalic Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Convert Blink to Italic option.

Syntax BlinkToItalic = TRUE/FALSE

Group Cfg3270 , Cfg5250 , CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.BlinkToItalic
HE.CurrentHost.Cfg3270.BlinkToItalic = TRUE
End Sub
```

Profile Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the profile name for the session. This method takes one parameter that is the name of the profile followed by the name of the folder separated by a period. For example, to set a profile called "VMCMS" in the "3270" folder, pass a string of "VMCMS.3270". You can also pass a complete path/file specification (including extension).

Syntax	Profile = szNewProfile$
Group	Cfg3270, Cfg5250, CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szValue = He.CurrentHost.Cfg3270.Profile HE.CurrentHost.Cfg3270.Profile = "ProfileName.FolderName" End Sub

BSIsDel CfgVT Property VT

You can use this property to get or set the Backspace Key option. When set (TRUE), the Backspace key will send a Del (127) code. When reset (FALSE), the Backspace key sends a BS (8) code.

Syntax BSIsDel = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.BSIsDel
HE.CurrentHost.CfgVT.BSIsDel = TRUE
End Sub
```

ProportionalFonts Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Proportional Fonts option.

Syntax ProportionalFonts = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.ProportionalFonts
HE.CurrentHost.Cfg3270.ProportionalFonts = TRUE
End Sub
```

ClearAllTabStops Cfgxxxx Method 3270 5250 VT

You can use this method to clear all tab stops.

Syntax	ClearAllTabStops
Group	Cfg3270 , Cfg5250 , CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Cfg3270.ClearAllTabStops End Sub

Related Topics

TabStop Cfgxxxx Method

TabStop Cfgxxxx Method 3270 5250 VT

You can use this method to set a tabstop at a given column. The column value must be between 1 and the total number of columns.

Syntax	TabStop iColumn%
Group	Cfg3270, Cfg5250, CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Cfg3270.TabStop 23 End Sub

Related Topics

ClearAllTabStops Cfgxxxx Method

RawAddFormFeed Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Add FormFeed after Raw Print Screen option.

Syntax RawAddFormFeed = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.RawAddFormFeed
HE.CurrentHost.Cfg3270.RawAddFormFeed = TRUE
End Sub
```

ClearScreenOnSizeChange CfgVT Property VT

You can use this property to get or set the Clear Screen on Size Change option. When set (TRUE), the screen is automatically cleared when the screen dimension changes (rows and/or columns). When reset (FALSE), the screen is not cleared when the dimensions change.

Syntax ClearScreenOnSizeChange = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.ClearScreenOnSizeChange
HE.CurrentHost.CfgVT.ClearScreenOnSizeChange = TRUE
End Sub
```

RawCaptureMode CfgVT Property VT

You can use this property to get or set the Capture Mode option. When set to TRUE and capturing is enabled, HostExplorer captures the raw data as it receives it. When reset to FALSE and capturing is enabled, HostExplorer captures only lines that are terminated by LF, VT, or FF.

Syntax RawCaptureMode = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.RawCaptureMode
HE.CurrentHost.CfgVT.RawCaptureMode = TRUE
End Sub
```

ColorDisplay Cfg5250 Property 5250

You can use this property to get or set the Color Display option for a 5250 terminal. This determines whether HostExplorer will use the values in the color attributes as a color map (TRUE) or as highlighting options (FALSE).

Syntax	ColorDisplay = TRUE/FALSE
Group	Cfg5250
Mode	5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg5250.ColorDisplay HE.CurrentHost.Cfg5250.ColorDisplay = TRUE End Sub

ReplyOEM Cfg3270 Property 3270

You can use this property to get or set the Send OEM Reply to RPQ option.

Syntax ReplyOEM = TRUE/FALSE

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.ReplyOEM
HE.CurrentHost.Cfg3270.ReplyOEM = TRUE
End Sub
```

CompressBlankLinesInScrollback CfgVT Property VT

You can use this property to get or set the Compress Blank Lines In Scrollback option. When enabled, HostExplorer does not add blank lines to the scrollback buffer. Changing this option does not affect the current contents of the scrollback buffer.

Syntax CompressBlankLinesInScrollback = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.CompressBlankLinesInScrollback
HE.CurrentHost.CfgVT.CompressBlankLinesInScrollback = TRUE
End Sub
```

ReRunAutoMacro Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the ReRun Auto Macro option.

Syntax ReRunAutoMacro = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.ReRunAutoMacro
HE.CurrentHost.Cfg3270.ReRunAutoMacro = TRUE
End Sub
```

RespectNumeric Cfg3270 Property 3270

You can use this property to get or set the Respect Numeric Fields option.

Syntax RespectNumeric = TRUE/FALSE

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.RespectNumeric
HE.CurrentHost.Cfg3270.RespectNumeric = TRUE
End Sub
```

ConnectTimeout Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the connect-timeout option. Use this option when establishing a new session.

Syntax ConnectTimeout = iNewTimeout%

Group Cfg3270 , Cfg5250 , CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.ConnectTimeout
HE.CurrentHost.Cfg3270.ConnectTimeout = 60
End Sub
```

ConvertNulls Cfg3270 Property 3270

You can use this property to get or set the Convert Nulls to Blanks option.

Syntax ConvertNulls = TRUE/FALSE

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.ConvertNulls
HE.CurrentHost.Cfg3270.ConvertNulls = TRUE
End Sub
```

SaveAppend Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Save Mode Append option.

Syntax SaveAppend = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.SaveAppend
HE.CurrentHost.Cfg3270.SaveAppend = TRUE
End Sub
```

SaveAttrsInScrollback CfgVT Property VT

You can use this property to get or set the Save Attrs In Scrollback option.

Syntax SaveAttrsInScrollback = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.SaveAttrsInScrollback
HE.CurrentHost.CfgVT.SaveAttrsInScrollback = TRUE
End Sub
```

CRToCRLF CfgVT Property VT

You can use this property to get or set the CR to CR/LF option. When set to TRUE, the return key will send a CR/LF sequence. When reset to FALSE, the return key sends CR.

Syntax CRToCRLF = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.CRToCRLF
HE.CurrentHost.CfgVT.CRToCRLF = TRUE
End Sub
```

SaveConfirm Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Confirm All Saves option.

Syntax SaveConfirm = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.SaveConfirm
HE.CurrentHost.Cfg3270.SaveConfirm = TRUE
End Sub
```

CursorKeyMode CfgVT Property VT

You can use this property to get or set the Cursor Key Mode. When set to TRUE, the cursor key is in Application Mode. When reset to FALSE, the cursor keys operate in normal mode.

Syntax	CursorKeyMode = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.CfgVT.CursorKeyMode HE.CurrentHost.CfgVT.CursorKeyMode = TRUE End Sub

Related Topics

KeypadMode CfgVT Property

KeypadMode CfgVT Property VT

You can use this property to get or set the Keypad mode. With a value of TRUE, the keypad operates in Application mode. When reset to FALSE, the keypad operates in Numeric mode.

Syntax	KeypadMode = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.CfgVT.KeypadMode HE.CurrentHost.CfgVT.KeypadMode = TRUE End Sub

Related Topics

CursorKeyMode CfgVT Property

CursorMode Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Cursor Mode option. The values can be CURSOR_BLINK or CURSOR_SOLID.

Syntax	CursorMode = iNewMode%
Group	Cfg3270 , Cfg5250 , CfgVT
Mode	3270, 5250, VT
Example	'$include:"-E\hebasic.ebh" Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.Cfg3270.CursorMode HE.CurrentHost.Cfg3270.CursorMode = CURSOR_BLINK End Sub

Related Topics

CursorType Cfgxxxx Property

CursorType Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Cursor Type option. The value can be CURSOR_UNDERSCORE, CURSOR_BLOCK, or CURSOR_LINE.

Syntax CursorType = iNewType%

Group CfgVT

Mode 3270, 5250, VT

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )
iValue = HE.CurrentHost.Cfg3270.CursorType
HE.CurrentHost.Cfg3270.CursorType = CURSOR_BLOCK
End Sub
```

Related Topics

CursorMode Cfgxxxx Property

SaveFontOnExit Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Save Font Information on Exit option.

Syntax SaveFontOnExit = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.SaveFontOnExit
HE.CurrentHost.Cfg3270.SaveFontOnExit = TRUE
End Sub
```

CursorSelectMode Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Cursor Select Mode option. The value can be CURSOR_RUBBERBAND or CURSOR_HIGHLIGHT.

Syntax CursorSelectMode = iMode%

Group CfgVT

Mode 3270, 5250, VT

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.CursorSelectMode
HE.CurrentHost.Cfg3270.CursorSelectMode = CURSOR_RUBBERBAND
End Sub
```

SaveMode Cfgxxxx Property 3270 5250 VT

You can use this property to set the Save As option. You can set the value as SAVE_ASCII or SAVE_ANSI.

Syntax SaveMode = iNewMode%

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
'$include:"-E\hebasci.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.SaveMode
HE.CurrentHost.Cfg3270.SaveMode = SAVE_ANSI
End Sub
```

SaveProfile Cfgxxxx Method 3270 5250 VT

You can use this method to save the current session settings to a profile on disk. This method takes one parameter that is the name of the profile followed by the name of the folder separated by a period. For example, to save a profile called "VMCMS" in the "3270" folder, pass a string of "VMCMS.3270". You can also pass a complete path/file specification (including extension).

Syntax	SaveProfile szProfileName$
Group	Cfg3270, Cfg5250, CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Cfg3270.SaveProfile "ProfileName.FolderName" End Sub

SaveProfileOnClose Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Save Profile on Window Close option.

Syntax SaveProfileOnClose = TRUE/FALSE

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.SaveProfileOnClose
HE.CurrentHost.Cfg3270.SaveProfileOnClose = TRUE
End Sub
```

DefaultHeight CfgVT Property VT

You can use this property to get or set the Default Height of the session. The possible values are 24 to 72.

Syntax	DefaultHeight = iNewValue%
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.CfgVT.DefaultHeight HE.CurrentHost.CfgVT.DefaultHeight = 24 End Sub

Related Topics

DefaultWidth CfgVT Property

DefaultWidth CfgVT Property VT

You can use this property to get or set the Default Width of the session. The possible values are 80 to 200.

Syntax DefaultWidth = iNewValue%

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

Value = HE.CurrentHost.CfgVT.DefaultWidth
HE.CurrentHost.CfgVT.DefaultWidth = 80
End Sub
```

Related Topics

DefaultHeight CfgVT Property

ShowHotspots Cfgxxx Property 3270 5250 VT

You can use this property to get or set the Show Hotspots option.

Syntax	ShowHotspots = TRUE/FALSE
Group	Cfg3270, Cfg5250, CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg3270.ShowHotspots HE.CurrentHost.Cfg3270.ShowHotspots = TRUE End Sub

Dev7171Terminal Cfg3270 Property 3270

You can use this property to get or set the 7171 Passthru Printing terminal type option. The value can be DEV7171_VT100, or DEV7171_IBM3164.

Syntax Dev7171Terminal =iNewTermType%

Group Cfg3270

Mode 3270

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.Dev7171Terminal
HE.CurrentHost.Cfg3270.Dev7171Terminal = DEV7171_VT100
End Sub
```

ShowNulls Cfgxxxx Property 3270 5250

You can use this property to get or set the Show Nulls option.

Syntax ShowNulls = TRUE/FALSE

Group Cfg3270, Cfg5250

Mode 3270, 5250

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.ShowNulls
HE.CurrentHost.Cfg3270.ShowNulls = TRUE
End Sub
```

Display3DBorder Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Display 3D Borders option.

Syntax Display3DBorder = TRUE/FALSE

Group CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.Display3DBorder
HE.CurrentHost.Cfg3270.Display3DBorder = TRUE
End Sub
```

SmoothScrolling CfgVT Property VT

You can use this property to get or set the Smooth Scrolling option. When set (TRUE), HostExplorer scrolls data using a smooth scroll method. When reset to FALSE, HostExplorer scrolls data line-by-line.

Before setting the SmoothScrolling Statement, activate the CfgVT.OptimizedDisplayMode=FALSE statement and manually set the DisplayMode statement to Realistic.

Syntax SmoothScrolling = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.SmoothScrolling
HE.CurrentHost.CfgVT.SmoothScrolling = TRUE
End Sub
```

DisplayAttr Cfg3270 Property 3270

You can use this property to get or set the Display Attributes option.

Syntax DisplayAttr = TRUE/FALSE

Group Cfg3270

Mode 3270

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.DisplayAttr
HE.CurrentHost.Cfg3270.DisplayAttr = TRUE
End Sub
```

SmoothScrollSpeed CfgVT Property VT

You can use this property to get or set the Smooth Scrolling Speed option. The value can be from 1 to 30. If the value is greater than the number of vertical pixels per cell, then scrolling occurs in Linemode.

Syntax SmoothScrollSpeed = iNewValue%

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.CfgVT.SmoothScrollSpeed
HE.CurrentHost.CfgVT.SmoothScrollSpeed = 2
End Sub
```

DisplayControlCodes CfgVT Property VT

You can use this property to get or set the Display Control Codes option. When set to TRUE, the system displays control codes, which are received. When reset to FALSE, HostExplorer acts upon the control codes it receives.

Syntax DisplayControlCodes = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.DisplayControlCodes
HE.CurrentHost.CfgVT.DisplayControlCodes = TRUE
End Sub
```

Sound Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Sound option.

Syntax	Sound = TRUE/FALSE
Group	Cfg3270, Cfg5250, CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg3270.Sound HE.CurrentHost.Cfg3270.Sound = TRUE End Sub

DisplayRowCol Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Display Row/Col Indicator option.

Syntax DisplayRowCol = TRUE/FALSE

Group CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = He.CurrentHost.Cfg3270.DisplayRowCol
HE.CurrentHost.Cfg3270.DisplayRowCol = TRUE
End Sub
```

StatusLineMode CfgVT Property VT

You can use this property to get or set the Status Linemode option. You can use the following literals to test values:

· STATUS_NONE

· STATUS_HOSTWRITABLE

· STATUS_INDICATOR

Syntax StatusLineMode = iNewValue%

Group CfgVT

Mode VT

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.CfgVT.StatusLineMode
HE.CurrentHost.CfgVT.StatusLineMode = STATUS_INDICATOR
End Sub
```

DisplayUpperCase Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the All Upper Case option.

Syntax DisplayUpperCase = TRUE/FALSE

Group CfgVT

Mode 3270, 5250

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.DisplayUpperCase
HE.CurrentHost.Cfg3270.DisplayUpperCase = TRUE
End Sub
```

EntryAssist Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Entry Assist option.

Syntax	EntryAssist = TRUE/FALSE
Group	CfgVT
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg3270.EntryAssist HE.CurrentHost.Cfg3270.EntryAssist = TRUE End Sub

Related Topics

WordWrap Cfgxxxx Property

WordWrap Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Word Wrap option.

Syntax	WordWrap = TRUE/FALSE
Group	Cfg3270, Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg3270.WordWrap HE.CurrentHost.Cfg3270.WordWrap = TRUE End Sub

Related Topics

EntryAssist Cfgxxxx Property

TCPPort Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the TCP Port value. The default value is 23 for telnet.

Syntax TCPPort = iNewPort%

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.TCPPort
HE.CurrentHost.Cfg3270.TCPPort = 23
End Sub
```

TelnetEcho Cfgxxxx Property VT

You can use this property to get or set the Telnet Echo option. Possible values are:

· 0—No.

· 1—Yes.

· 2—Automatic.

Syntax	TelnetEcho = iNewEchoMode%
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.CfgVT.TelnetEcho HE.CurrentHost.CfgVT.TelnetEcho = 2 End Sub

Related Topics

LineMode CfgVT Property

Linemode CfgVT Property VT

You can use this property to get or set the Telnet Linemode option. This option is used during telnet option negotiation with a host. Possible values are:

· 0—Don't do linemode.

· 1—Always do linemode.

· 2—During Local Echo.

· 3—When Not in SGA.

· 4—When Local Echo or Not SGA.

Syntax	Linesmode = iLineMode%
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iValue = HE.CurrentHost.CfgVT.Linemode HE.CurrentHost.CfgVT.Linemode = 0 End Sub

Related Topics


TelnetEcho CfgVT Property

ForceAltSize Cfg3270 Property 3270

You can use this property to get or set the Force Alternate Size option.

Syntax ForceAltSize = TRUE/FALSE

Group Cfg5250

Mode 3270, 5250

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.ForceAltSize
HE.CurrentHost.Cfg3270.ForceAltSize = TRUE
End Sub
```

TelnetName Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Telnet Name Override option.

Syntax TelnetName = szNewName$

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue = HE.CurrentHost.Cfg3270.TelnetName
HE.CurrentHost.Cfg3270.TelnetName = "IBM-3278-2"
End Sub
```

ForceExactSize Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Force Exact Size Font option.

Syntax ForceExactSize = TRUE/FALSE

Group CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.Cfg3270.ForceExactSize
HE.CurrentHost.Cfg3270.ForceExactSize = TRUE
End Sub
```

TPRINTDestination Cfg3270 Property 3270

You can use this property to get or set the Send PCPRINT/TPRINT Output To option. Possible values are:

· TPRINT_DEFAULT

· TPRINT_CLIPBOARD

· TPRINT_LPT1

· TPRINT_LPT2

· TPRINT_LPT3

Syntax TPRINTDestination = iNewMode%

Group Cfg3270

Mode 3270

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.Cfg3270.TPRINTDestination
HE.CurrentHost.Cfg3270.TPRINTDestination = TPRINT_CLIPBOARD
End Sub
```

FTPApplicationName Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the FTP Application Name option.

Syntax FTPApplicationName = newFTPApplicationName$

Group CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szOldValue$ = HE.CurrentHost.Cfg3270.FTPApplicationName
HE.CurrentHost.Cfg3270.FTPApplicationName = "C:\WS_FTP32.EXE"
End Sub
```

TypeAhead Cfgxxxx Property 3270 5250

You can use this property to retrieve or set the Type Ahead option.

Syntax	TypeAhead = TRUE/FALSE
Group	Cfg3270, Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bValue = HE.CurrentHost.Cfg3270.TypeAhead HE.CurrentHost.Cfg3270.TypeAhead = TRUE End Sub

Related Topics

Host Keys Method

Host RunCmd Method

Host Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the IP Host/Gateway option (for Telnet connections) or the phone number to dial (for Modem connections).

Syntax Host = szNewHost$

Group CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue$ = HE.CurrentHost.Cfg3270.Host
HE.CurrentHost.Cfg3270.Host = "myhost.mydomain"
End Sub
```

UPSSet CfgVT Property VT

You can use this property to get or set the User Preferred Supplemental Character Set. The following string values can be set:

· "DEC Supplemental"

· "ISO Latin-1 (8859-1)"

· "PC English (437)"

· "Europa3 (333)"

· "Nomos (373)"

· "ISO Latin-9 (8859-15)"

Syntax UPSSet = iNewValue$

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue = HE.CurrentHost.CfgVT.UPSSet
HE.CurrentHost.CfgVT.UPSSet = "DEC Supplemental"
End Sub
```

KermitBinPrefix CfgVT Property VT

You can use this property to get or set the Binary Prefix flag.

Syntax	KermitBinPrefix = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.KermitBinPrefix HE.CurrentHost.CfgVT.KermitBinPrefix = FALSE End Sub

Related Topics

KermitCompression CfgVT Property

KermitTextMode CfgVT Property

KermitUseFullPath CfgVT Property

KermitCompression CfgVT Property VT

You can use this property to get or set the RLE Compression flag.

Syntax	KermitCompression = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.KermitCompression HE.CurrentHost.CfgVT.KermitCompression = FALSE End Sub

Related Topics

KermitBinPrefix CfgVT Property

KermitTextMode CfgVT Property

KermitUseFullPath CfgVT Property

KermitTextMode CfgVT Property VT

You can use this property to get or set the Text Mode flag.

Syntax	KermitTextMode = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.KermitTextMode HE.CurrentHost.CfgVT.KermitTextMode = FALSE End Sub

Related Topics

KermitBinPrefix CfgVT Property

KermitCompression CfgVT Property

KermitUseFullPath CfgVT Property

KermitUseFullPath CfgVT Property VT

You can use this property to get or set the Use Full Path Name to/from Host flag.

Syntax	KermitUseFullPath = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.KermitUseFullPath HE.CurrentHost.CfgVT.KermitUseFullPath = FALSE End Sub

Related Topics

KermitBinPrefix CfgVT Property

KermitCompression CfgVT Property

KermitTextMode CfgVT Property

XModem16BitCrc CfgVT Property VT

You can use this property to get or set the 16-bit CRC flag. This enables 16-bit Cyclic Redundancy Checksum for XModem file transfers.

Syntax XModem16BitCrc = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bVal = HE.CurrentHost.CfgVT.XModem16BitCrc
HE.CurrentHost.CfgVT.XModem16BitCrc = TRUE/FALSE
End Sub
```

Related Topics

XModemPkt1024 CfgVT Property

XModemSendTimeout CfgVT Property

XModemPkt1024 CfgVT Property VT

You can use this property to get or set the 1024-byte packet flag. This enables XModem to use 1024-byte packets if the host software supports this option.

Syntax XModemPkt1024 = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bVal = HE.CurrentHost.CfgVT.XModemPkt1024
HE.CurrentHost.CfgVT.XModemPkt1024 = TRUE/FALSE
End Sub
```

Related Topics

XModem16BitCrc CfgVT Property

XModemSendTimeout CfgVT Property

XModemSendTimeout CfgVT Property VT

You can use this property to get or set the send timeout for XModem file transfers. The time set is in milliseconds.

Syntax XModemSendTimeout = lTimeout&

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

lVal& = HE.CurrentHost.CfgVT.XModemSendTimeout
HE.CurrentHost.CfgVT.XModemSendTimeout = 15000
End Sub
```

Related Topics

XModem16BitCrc CfgVT Property

XModemPkt1024 CfgVT Property

KeyboardProfileName Cfgxxxx Method 3270 5250 VT

You can use this method to change the current keyboard profile. This method loads the specified keyboard profile as the active session keyboard profile.

Syntax KeyboardProfileName szKeyProfile$

Group CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

HE.CurrentHost.Cfg3270.KeyboardProfileName = "MYKEYBOARD"
End Sub
```

YModemSendTimeout CfgVT Property VT

You can use this property to get or set the send timeout for Y-Modem file transfers. The time set is in milliseconds.

Syntax YModemSendTimeout = lTimeout&

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

lVal& = HE.CurrentHost.CfgVT.YModemSendTimeout
HE.CurrentHost.CfgVT.YModemSendTimeout = 15000
End Sub
```

Related Topics

YModemUseFullPath CfgVT Property

YModemUseFullPath CfgVT Property VT

You can use this property to get or set the Use Full Path Name to/from Host flag.

Syntax	YModemUseFullPath = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.YModemUseFullPath HE.CurrentHost.CfgVT.YModemUseFullPath = FALSE End Sub

Related Topics

YModemSendTimeout CfgVT Property

LinesInScrollback CfgVT Property VT

You can use this property to get or set the Lines Attrs In Scrollback option. This is the number of lines available in the Scrollback buffer. Use a value of 0 to disable the scrollback buffer.

Syntax LinesInScrollback = iNewLines%

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iValue = HE.CurrentHost.CfgVT.LinesInScrollback
HE.CurrentHost.CfgVT.LinesInScrollback= 120
End Sub
```

ZModemAutoDownload CfgVT Property VT

You can use this property to get or set the Auto Download flag.

Syntax	ZModemAutoDownload = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.ZModemAutoDownload HE.CurrentHost.CfgVT.ZModemAutoDownload = FALSE End Sub

Related Topics

ZModemCrashRecovery CfgVT Property

ZModemMaxErr CfgVT Property

ZModemOverwrite CfgVT Property

ZModemSlidingBytes CfgVT Property

ZModemSlidingWin CfgVT Property

ZModemUseFullPath CfgVT Property

ZModemCrashRecovery CfgVT Property VT

You can use this property to get or set the Crash Recovery flag.

Syntax ZModemCrashRecovery = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bVal = HE.CurrentHost.CfgVT.ZModemCrashRecovery
HE.CurrentHost.CfgVT.ZModemCrashRecovery = FALSE
End Sub
```

Related Topics

ZModemAutoDownload CfgVT Property

ZModemMaxErr CfgVT Property

ZModemOverwrite CfgVT Property

ZModemSlidingBytes CfgVT Property

ZModemSlidingWin CfgVT Property

ZModemUseFullPath CfgVT Property

ZModemMaxErr CfgVT Property VT

You can use this property to get or set the Crash Recovery value.

Syntax	ZModemMaxErr = iVal
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iVal = HE.CurrentHost.CfgVT.ZModemMaxErr HE.CurrentHost.CfgVT.ZModemMaxErr = 15 End Sub

Related Topics

ZModemAutoDownload CfgVT Property

ZModemCrashRecovery CfgVT Property

ZModemOverwrite CfgVT Property

ZModemSlidingBytes CfgVT Property

ZModemSlidingWin CfgVT Property

ZModemUseFullPath CfgVT Property

ZModemOverwrite CfgVT Property VT

You can use this property to get or set the Overwrite Management Option. Possible values are:

· Z_ALWAYS

· Z_APPEND

· Z_NEWER_LONGER

· Z_NEWER

· Z_NEVER

· Z_SIZE_DATE_DIFFER

Syntax ZModemOverwrite = iVal%

Group CfgVT

Mode VT

Example

```
'$include:"-E\hebasic.ebh"

Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

iVal% = HE.CurrentHost.CfgVT.ZModemOverwrite
HE.CurrentHost.CfgVT.ZModemOverwrite = Z_ALWAYS
End Sub
```

Related Topics

ZModemAutoDownload CfgVT Property

ZModemCrashRecovery CfgVT Property

ZModemMaxErr CfgVT Property

ZModemSlidingBytes CfgVT Property

ZModemSlidingWin CfgVT Property

ZModemUseFullPath CfgVT Property

ZModemSlidingBytes CfgVT Property VT

You can use this property to get or set the Sliding Window Size value.

Syntax	ZModemSlidingBytes = iVal
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") iVal = HE.CurrentHost.CfgVT.ZModemSlidingBytes HE.CurrentHost.CfgVT.ZModemSlidingBytes = 8192 End Sub

Related Topics

ZModemAutoDownload CfgVT Property

ZModemCrashRecovery CfgVT Property

ZModemMaxErr CfgVT Property

ZModemOverwrite CfgVT Property

ZModemSlidingWin CfgVT Property

ZModemUseFullPath CfgVT Property

ZModemSlidingWin CfgVT Property VT

You can use this property to get or set the Sliding Window flag.

Syntax	ZModemSlidingWin = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.ZModemSlidingWin HE.CurrentHost.CfgVT.ZModemSlidingWin = FALSE End Sub

Related Topics

ZModemAutoDownload CfgVT Property

ZModemCrashRecovery CfgVT Property

ZModemMaxErr CfgVT Property

ZModemOverwrite CfgVT Property

ZModemSlidingBytes CfgVT Property

ZModemUseFullPath CfgVT Property

ZModemUseFullPath CfgVT Property VT

You can use this property to get or set the Use Full Path Name to/from Host flag.

Syntax	ZModemUseFullPath = TRUE/FALSE
Group	CfgVT
Mode	VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") bVal = HE.CurrentHost.CfgVT.ZModemUseFullPath HE.CurrentHost.CfgVT.ZModemUseFullPath = FALSE End Sub

Related Topics

ZModemAutoDownload CfgVT Property

ZModemCrashRecovery CfgVT Property

ZModemMaxErr CfgVT Property

ZModemOverwrite CfgVT Property

ZModemSlidingBytes CfgVT Property

ZModemSlidingWin CfgVT Property

LoadProfile Cfgxxxx Method 3270 5250 VT

You can use this method to load a saved profile into the current session. There are several ways to specify the profile path.

· If the profile file is in the Profile directory where the user files are stored on your machine, specify the relative path to this directory. For example: DefaultVT\My Profile.hep.

Note: Specifying the .hep extension is optional.

· You can specify the full directory path and file name of the profile file. If the file is in the Profile directory where your user files are stored on your machine, specify the full path. For example: C:\WINNT\Profiles\All Users\Application Data\Hummingbird\

Connectivity\7.10\Profile\My Profile.hep. For the appropriate directory path for your platform, refer to the list of default locations for the user files. 

Note: Specifying the .hep extension is optional.

· For compatibily with HostExplorer versions prior to V7.1, you can specify a string that consists of the profile name (without the .hep extension) followed by the name of the folder separated by a period. For example: My Profile.DefaultVT.

Syntax	LoadProfile szProfileName$
Group	CfgVT
Mode	3270, 5250, VT
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") HE.CurrentHost.Cfg3270.LoadProfile _ DefaultVT\My Profile End Sub

LocalEcho CfgVT Property VT

You can use this property to get or set the Local Echo option. When set to TRUE, HostExplorer echoes all data entered. When reset to FALSE, HostExplorer does not echo any data entered.

Syntax LocalEcho = TRUE/FALSE

Group CfgVT

Mode VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

bValue = HE.CurrentHost.CfgVT.LocalEcho
HE.CurrentHost.CfgVT.LocalEcho = TRUE
End Sub
```

LongName Cfgxxxx Property 3270 5250 VT

You can use this property to get or set the Session Long Name option. This property will accept only the first 8 characters.

Syntax LongName = szNewName$

Group Cfg3270, Cfg5250, CfgVT

Mode 3270, 5250, VT

Example

```
Sub Main
Dim HE as Object
Set HE = CreateObject( "HostExplorer" )

szValue = HE.CurrentHost.Cfg3270.LongName
HE.CurrentHost.Cfg3270.LongName = "VM/CMS"
End Sub
```

LUName Cfgxxx Property 3270 5250

You can use this property to get or set the LU (Logical Units) Name option.

Syntax	LUName = szNewLUName$
Group	Cfg3270, Cfg5250
Mode	3270, 5250
Example	Sub Main Dim HE as Object Set HE = CreateObject("HostExplorer") szValue = HE.CurrentHost.Cfg3270.LUName HE.CurrentHost.Cfg3270.LUName = "LU1234" End Sub

EHLLAPI and WinHLLAPI DLL Support

HostExplorer supports the multiple HLLAPI (High Level Language Application Programming Interface) dynamic-link libraries (DLLs) for complete compatibility with Attachmate® Extra! for Windows. These interfaces allow other Windows programs to communicate and control the 3270 and 5250 emulators and partially control the Telnet emulator.

Note: Unless specified explicitly, the term HLLAPI refers to all supported DLLs.

The HLLAPI DLLs are contained in the following modules:

· EHLLAPI (Extended HLLAPI) module

· WinHLLAPI (Windows HLLAPI) module

The EHLLAPI interface includes:

· a new Window Close (201) function 

· an extended ConnectPS(1) function 

Related Topics

Irma Compatibility Mode

Configuration Tips

EHLLAPI Module

The EHLLAPI module (16-Bit=ACS3EHAP.DLL, 32-Bit=EHLLAP32.DLL, EHLAPI32.DLL) is compatible with Attachmate® Extra! for Windows. Because most vendors' products support multiple HLLAPI DLLs, always choose Attachmate® Extra! for Windows EHLLAPI. Check the Compatibility option to ensure that the emulator you are using is compatible. This interface is available with both the 16-bit and the 32-bit versions of HostExplorer.

Note: If you choose an Irma Workstation for Windows setting, make sure to set the Irma compatibility in the EHLLAPI dialog box.

Related Topics

EHLLAPI Development Files

EHLLAPI Calls

EHLLAPI Development Files

To develop new applications that need to communicate with the 3270 or 5250 emulator, use the WinHLLAPI interface. This is the only standardized interface for 16-bit and 32-bit platforms. The development and online documentation files are installed on your system.

The following EHLLAPI development files are available:

· ACS3EHAP.H—EHLLAPI 'C' header file

· ACS3EHAP.LIB—EHLLAPI 16-bit link library

· EHLLAP32.LIB—EHLLAPI 32-bit link library

· ACS3EHAP.DLL—EHLLAPI DLL (16-bit)

· EHLLAP32.DLL—EHLLAPI DLL (32-bit)

· EHLLTEST.EXE—EHLLAPI Interactive test program (16-bit)

Related Topics

Special HLLAPI and WinHLLAPI Flags

EHLLAPI Support in VT and NVT Modes

WinHLLAPI Development Files

Special EHLLAPI and WinHLLAPI Flags

When the HLLAPI spawns a new session automatically by starting a profile, it may have trouble synchronizing with the initial Host Logon panel. Although the TCP/IP connection is complete, it may take extra time for the host to paint the logon panel (HostExplorer waits for the first host update). You may need to insert an additional wait before the ConnectPS actually returns.

The following special EHLLAPI flags are available:

Auto Start Delay

Start Minimized

Auto Unload

Yield Wait

Return Extra Session Info

Auto Sync

Allow Connect Physical

Convert Nulls

Update Screen After Copy

Related Topics

EHLLAPI Support in VT and NVT Mode

EHLLAPI Development Files

NVT Mode Functions

NVT Mode Exceptions

Auto Start Delay

To add an additional wait command:

1. Launch HostExplorer, then establish a remote host connection.

2. Add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

[EHLLAPI.Settings]

Auto Start Delay = x

This delays the ConnectPS return after the connection is complete.

3. Replace x with the number of seconds.

Note: x is the number of seconds you want the system to wait. By default, this value is 1.

Start Minimized

When HLLAPI spawns a new session automatically by starting a profile, that window is kept hidden by the emulator because it is under HLLAPI control.

To force newly spawned sessions in minimized mode (iconized and visible), add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]
Start Minimized = On
```

Auto Unload

When you issue a DisconnectPS, HLLAPI terminates that terminal session automatically if the session was spawned by HLLAPI.

To prevent HLLAPI from terminating the session, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

```
[EHLLAPI.Settings]
Auto Unload = Off
```

Yield Wait

By default, the functions that require the DLL to wait for some event to complete (such as Wait, Pause, Send File, and Receive File) use a PeekMessage loop in order to let all applications process messages. However, you can set a loop call to yield the wait.

To set a loop call, set the following line in the EHLLAPI.Settings section of the HOSTEX.INI file:

```
[EHLLAPI.Settings]
Yield Wait = On
```

Return Extra Session Info

To set the last byte of the 18-byte structure:

1. Add one of the following lines to the EHLLAPI.Settings section in the HOSTEX.INI file:

'I' — Idle - Configured but not loaded and not connected.

'R' — Ready - Session connected to host but not connected to HLLAPI.

'C' — Connect - Session connected to host and connected to HLLAPI (Connect PS).

This sets the last byte of the 18-byte structure, normally reserved to a flag providing this information.

2. Add the following lines to the EHLLAPI.Settings section in the HOSTEX.INI file:

[EHLLAPI.Settings]

Return Extra Session Info = On

This enables this extra flag byte.

Note: By default, the HLLAPI standard interface does not provide any mechanism to know whether a session is: connected (with HLLAPI), loaded and connected to a host but not to HLLAPI, or simply configured and not connected at all.

Auto Sync

The SendKey function (in its current design) does not allow for automatic pacing when you press AID generating keys. Therefore, if you want to send two sets of strings in a row, such as *XYZ@E*, you must place a WAIT(TWAIT) command between them. You can instruct HLLAPI to wait until the keyboard unlocks before returning from the SendKey function when you press an AID key. This extension provides an automatic synchronization with the host and simplifies your HLLAPI application.

To enable Auto Sync:

1. Enable the Type Ahead feature in the profile assigned to the HLLAPI short name. You can set the Type Ahead feature by double-clicking the Session folder and clicking General in the Session Profile dialog box.

2. Add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

[EHLLAPI.Settings]

Auto Sync = On

Allow Connect Physical

According to common specifications, the CONPHYS flag that is used in DOS to perform a physical connect (bring window to front) is not supported in EHLLAPI (ACS3EHAP.DLL).

To bring the window to the front, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

[EHLLAPI.Settings]
Allow Connect Physical= On

Convert Nulls

The CopyPS and CopyPSToString functions normally convert 3270/5250 Nulls to ASCII blanks when you copy text.

To prevent HLLAPI from converting nulls, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file.

```
[EHLLAPI.Settings]
Convert Nulls = Off
```

Note: If you disable the Convert Nulls HLLAPI feature, you must use the STRLEN option to use explicit string lengths or change the EOT character from the default value of Null.

Update Screen After Copy

This option forces the emulator to repaint the screen when an HLLAPI application copies data to the screen buffer using the CopyString to PS and CopyStringToField functions.

To force the emulator to repaint the screen, add the following line to the EHLLAPI.Settings section in the HOSTEX.INI file:

[EHLLAPI.Settings]
Update Screen After Copy = On

Note: This function dramatically reduces performance. Enable this option for debugging purposes only.

EHLLAPI Support in VT and NVT Modes

HostExplorer supports the EHLLAPI interface while in VT and NVT (Network Virtual Terminal), or ANSI terminal modes. This allows your application to interact with a front-end system without requiring a new interface to work with the ANSI-terminal portion of the emulator.

While in NVT mode, only a subset of the EHLLAPI functions are supported. This is mainly because the NVT screen does not contain 3270 information and does not have a concept of fields.

EHLLAPI no longer supports extended return codes for NVT mode. To enable support for extended return codes, add the following line in the EHLLAPI.Settings section in the HOSTEX.INI file:

[EHLLAPI.Settings]
Enhanced RC = On

Related Topics

Special EHLLAPI and WinHLLAPI Flags

EHLLAPI Development Files

NVT Mode Functions

NVT Mode Exceptions

WinHLLAPI Development Files

To develop new applications that need to communicate with the 3270 or 5250 emulator, use the WinHLLAPI interface. This is the only standardized interface for 16-bit and 32-bit platforms. The development and online documentation files are installed on your system.

The following WinHLLAPI development files are available:

· WHLLAPI.HLP—WinHLLAPI on-line Help

· WHLLAPI.H—WinHLLAPI 'C' header file

· WHLLAPI.LIB—WinHLLAPI 16-bit link library

· WHLLAP32.LIB—WinHLLAPI 32-bit link library

· WHLLAPI.DLL—WinHLLAPI DLL (16-bit)

· WHLLAP32.DLL—WinHLLAPI DLL (32-bit)

· WHLLTEST.EXE—WinHLLAPI interactive test program (16-bit)

· WHLTST32.EXE—WinHLLAPI interactive test program (32-bit)

Related Topics

EHLLAPI Development Files

NVT Mode Functions

The following functions are supported normally while in NVT mode:

· Connect Presentation Space (1)

· Convert Position or RowCol (99)

· Disconnect Presentation Space (2)

· Pause (18)

· Query Close Intercept (42)

· Query Cursor Location (7)

· Query Sessions (10)

· Query System (20)

· Release (12)

· Reserve (11)

· Reset System (21)

· Search Presentation Space (6)

· Set Session (9)

· Start Close Intercept (41)

· Start Host Notification (23)

· Stop Close Intercept (43)

· Stop Host Notification (25)

Related Topics

Special EHLLAPI and WInHLLAPI Flags

EHLLAPI Development Files

EHLLAPI Support in VT and NVT Modes

NVT Mode Exceptions

The following functions perform differently while in NVT mode. The host may enter and exit NVT mode during your session, so do not assume that you will be in NVT mode only at the beginning of a session. Your application must be able to handle both modes.

Copy OIA (13)—The return value in the data_string is slightly different to let you determine whether the terminal is in NVT mode. Byte 82, labeled On-line and screen ownership (group 1), has the 0x01 bit on if the terminal is in NVT mode. This bit is normally reserved.

Copy Presentation Space (5) and Copy Presentation Space to String (8)—The return code in the ps_position value contains a 101 (decimal) if the terminal is in NVT mode. If the terminal is in 3270 mode, the standard values of 0, 1, 4, 5, and 9 apply.

Query Host Update (24)—The return code in the ps_position value contains a 101 (decimal) if the terminal is in NVT mode and the PS has not changed. It contains a 102 (decimal) if the terminal is in NVT mode and the PS has changed since the last time the query host was updated. If the terminal is in 3270 mode, the standard values of 0, 1, 8, 9, 21, 22, and 23 apply.

Query Session Status (22)—The return value in the data_string is slightly different to let you determine whether the terminal is in NVT mode. Byte 11, which provides the session characteristics, has the 0x01 bit on if the terminal is in NVT mode. This bit is normally reserved.

Send Key (3)—This function does not support any of the standard keyboard mnemonics except @E–(carriage return) and @D–(backspace).

Note: Do not send more than one carriage return per operation. Use the Query Host Update between CRs to synchronize with the host.

Related Topics

Special EHLLAPI and WinHLLAPI Flags

EHLLAPI Development Files

EHLLAPI Support in VT and NVT Modes

EHLLAPI Calls

When you program in Visual Basic, you can control the emulator by using the OLE Automation interface (recommended) or the EHLLAPI/WinHLLAPI interface. If you use the EHLLAPI/WinHLLAPI interface, include the HLLCALLS.TXT file (which is in the DEVKITS directory) in your project.

Edit the top of the HLLCALLS.TXT file to call either EHLLAPI or WinHLLAPI. If you use WinHLLAPI, remember that you must call EHLLAPIStartup before calling any EHLLAPI function and call EHLLAPICleanup as the last call in your program.

The following EHLLAPI calls are available with HostExplorer:

EHLLAPIConnect
EHLLAPIConvertPosToRowCol
EHLLAPIConvertRowColToPos
EHLLAPICopyFieldToString
EHLLAPICopyOIA
EHLLAPICopyPS
EHLLAPICopyPSToString
EHLLAPICopyStringToField
EHLLAPICopyStringToPS
EHLLAPIDisconnect
EHLLAPIFindFieldPosition
EHLLAPIGetRowString
EHLLAPIGetVersion
EHLLAPIPause
EHLLAPIQueryCursorLocation
EHLLAPIQueryFieldAttribute
EHLLAPIQuerySessions
EHLLAPIQuerySessionStatus
EHLLAPIReceiveFile
EHLLAPIRelease
EHLLAPIReserve
EHLLAPIReset
EHLLAPISearchField
EHLLAPISearchPS
EHLLAPISendFile
EHLLAPISendKey
EHLLAPISetCursorLocation
EHLLAPISetSessionParameters
EHLLAPIWait

Related Topics

EHLLAPI and WinHLLAPI DLL support

EHLLAPI Development Files

Visual Basic Interface

EHLLAPIConnect 3270 5250 VT

This function is used to download a file from the host system. The valid options are ASCII, CRLF, and APPEND. Separate all options with spaces, not commas.

Function

EHLLAPIReceiveFile (strPCFileName As String, idSession As String, StrHostFilename As String, strOptions As String) As Integer

Input

StrPCFileName—contains the name of the computerfile to receive the data.

IdSession—the session short-name identifier. It must be a single, uppercase letter.

StrHostFileName—contains the name of the HOST file to be downloaded.

StrOptions—contains the file-transfer options, separated by spaces.

Example

The following example transfers the host file PROFILE EXEC to the computer file C:\PROF.AUT:

```
IRc% = EHLLAPIReceiveFile( "C:\PROF.AUT", "A", "PROFILE EXEC",

"ASCII CRLF" )
```

EHLLAPIQueryFieldAttribute 3270 5250 VT

This function is used to get the attribute of the field at the location specified.

Function	EHLLAPIQueryFieldAttribute (iPos As Integer, iAttr As Integer) As Integer
Input	iPos—the presentation-space (PS) position to search for the field attribute.
Output	iAttr—the 3270 attribute returned.
Example	The following example retrieves the attribute at position 1920:

```
iRc% = EHLLAPIQueryFieldAttribute( 1920, iAttr% )
```

EHLLAPIConvertPosToRowCol 3270 5250 VT

This function is used to convert a presentation-space (PS) value to a row-and-column value.

Function	EHLLAPIConvertPositionToRowCol (idSession As String, iPos As Integer, iRow As Integer, iColumn As Integer) As Integer
Input	IdSession—the session short-name identifier. It must be a single, uppercase letter. IPos—the PS (1 to screen size) to be converted.
Output	IRow—the row position. IColumn—the column position.
Example	The following example converts the PS position 1761 on session "A" to a row-and-column value: Rc% = EHLLAPIConvertPositionToRowCol("A", 1761, iRow%, iCol%)

EHLLAPIQuerySessions 3270 5250 VT

This function returns a string containing the short names of all available sessions. For example, if the system returns an "AC" string, sessions "A" and "C" are available.

Function	EHLLAPIQuerySessions (strAllSessions As String) As Integer
Output	StrAllSessions—the VB string to receive the session short names.
Example	The following example retrieves the list of sessions.

```
iRc% = EHLLAPIQuerySessions( strAll$ )
```

EHLLAPIConvertRowColToPosition 3270 5250 VT

This function is used to convert a row-and-column value to a presentation-space (PS) value. Unlike other functions, the return code of this function is the new PS position.

Function	EHLLAPIConvertRowColToPosition (idSession As String, iRow As Integer, iColumn As Integer) As Integer
Input	IdSession—the session short-name identifier. It must be a single, uppercase letter. IRow—the row value between 1 and the maximum number of rows (typically 24). IColumn—the column value between 1 and the maximum number of columns (typically 80).
Output	IPos—the PS (1 to screen size) to be converted.
Example	The following example converts the row-and-column on session "A" value to a PS position: IPos% = EHLLAPIConvertRowColToPosition("A", 24, 1)

EHLLAPIQuerySessionStatus 3270 5250 VT

This function returns the session long name and the session screen size.

Function	EHLLAPIQuerySessionStatus (idSession As String, strLongName As String, IRows As Integer, iColumns As Integer) As Integer
Input	IdSession—the session short-name identifier. It must be a single, uppercase letter.
Output	StrLongName—contains the session long name (up to 8 characters). IRows—contains the number of rows in the presentation-space (PS). IColumns—contains the number of columns in the PS.
Example	The following example retrieves the info for session "A". iRc% = EHLLAPIQuerySessionStatus("A", strLongName$, iRows%, iCols%)

EHLLAPICopyFieldToString 3270 5250 VT

This function copies a field (or portion of a field) to a Visual Basic (VB) string. Attributes are translated to blanks, and no extended attributes are returned.

Function	EHLLAPICopyFieldToString (strString As String, iMaxLen As Integer, iPos As Integer) As Integer
Input	IPos—the presentation-space (PS) position (1 to the maximum screen size) of the field to copy. IMaxLen—the length of data to copy from iPos.
Output	StrString—the VB string to receive the field content. If the screen is unformatted, the result of this may be up to 3564 bytes.
Example	This example copies the contents of the field at position 81 for a maximum length of 80 bytes: IRc% = EHLLAPICopyFieldToString(strDest$, 80, 81)

EHLLAPIReceiveFile 3270 5250 VT

This function is used to download a file from the host. Valid options are ASCII, CRLF, and APPEND. Separate all options with spaces, not commas.

Function	EHLLAPIReceiveFile (strPCFileName As String, idSession As String, StrHostFilename As String, strOptions As String) As Integer
Input	StrPCFileName—contains the name of the computer file to receive the data. IdSession—the session short-name identifier. It must be a single, uppercase letter. StrHostFileName—contains the name of the HOST file to be downloaded. StrOptions—contains the file-transfer options, separated by spaces.
Example	The following example transfers the host file PROFILE EXEC to the computer file C:\PROF.AUT: iRc% = EHLLAPIReceiveFile("C:\PROF.AUT", "A", "PROFILE EXEC", "ASCII CRLF")

EHLLAPICopyOIA 3270 5250 VT

This function copies the Operator Information Area (OIA) to a string. The contents of this string are in the special EBCDIC format, as described in the *EHLLAPI Programming Guide*.

Function	EHLLAPICopyOIA (strOIA As String) As Integer
Output	StrOIA—the VB string to receive the OIA string. Length is always 103 bytes.
Example	This example copies the current OIA into a string:

```
iRc% = EHLLAPICopyOIA( strOIA$ )
```

EHLLAPIRelease 3270 5250 VT

This function is used to release the 3270 keyboard. Make this call only after a call to EHLLAPIReserve.

Function	EHLLAPIRelease () As Integer
Example	The following example releases the keyboard:

```
iRc% = EHLLAPIRelease()
```

EHLLAPICopyPS 3270 5250 VT

This function copies the entire presentation space (PS) to a Visual Basic (VB) string. Attributes are translated to blanks, and no extended attributes are returned.

Function	EHLLAPICopyPS (strScreen As String) As Integer
Output	StrScreen—the VB string to receive the screen image. The screen sizes are Model 2 (24x80) 1920, Model 3 (32x80) 2560, Model 4 (43x80) 3440, and Model 5 (27x132) 3564.
Example	This example copies the entire PS to a VB string: iRc% = EHLLAPICopyPS(strPS$)

EHLLAPIReserve 3270 5250 VT

This function is used to reserve the 3270 keyboard. This prevents the user from accessing the 3270 session from the keyboard.

Function	EHLLAPIReserve () As Integer
Example	The following example reserves the keyboard:

```
iRc% = EHLLAPIReserve()
```

EHLLAPICopyPSToString 3270 5250 VT

This function copies a portion of the presenation space (PS) to a Visual Basic (VB) string. Attributes are translated to blanks, and no extended attributes are returned.

Function	EHLLAPICopyPSToString (strString As String, iMaxLen As Integer, iPos As Integer) As Integer
Input	iPos—the PS position (1 to the maximum screen size) to start copying. iMaxLen—the length of data to copy from iPos.
Output	StrString—the VB string to receive the partial screen image.
Example	This example copies the PS space at starting at position 1700 for 220 bytes: iRc% = EHLLAPICopyPS(strPartial$, 220, 1700)

EHLLAPIReset 3270 5250 VT

This function is used to reset the EHLLAPI interface back to the default values.

Function	EHLLAPIReset () As Integer
Example	The following example resets the interface:

```
iRc% = EHLLAPIReset()
```

EHLLAPICopyStringToField 3270 5250 VT

This function copies a Visual Basic (VB) string to a 3270 field. Attributes are translated to blanks, and no extended attributes are returned.

Function	EHLLAPICopyStringToField (strString As String, iPos As Integer) As Integer
Input	StrString—the VB string to copy to the 3270 field.
	iPos—the PS position (1 to the maximum screen size) to which the information is copied.
Example	This example copies a string to position 1761:
	iRc% = EHLLAPICopyStringToField("Hello World", 1761)

EHLLAPISearchField 3270 5250 VT

This function is used to search a field for a given string.

Function	EHLLAPISearchField (strString As String, iPos As Integer) As Integer
Input	StrString—the Visual Basic (VB) string to search. iPos—the presentation-space (PS) position to begin the search.
Output	iPos—the PS position of the text, if found (if iRc =0).
Example	The following example searches for the string "More..." in the field, starting at position 1841: iPos% = 1841 iRc% = EHLLAPISearchField("More...", iPos%)

EHLLAPICopyStringToPS 3270 5250 VT

This function copies a string to the presentation space (PS) at the location specified. Attributes are translated to blanks, and no extended attributes are returned.

Function	EHLLAPICopyStringToPS (strString As String, iPos As Integer) As Integer
Input	StrString—the Visual Basic (VB) string to copy to the PS.
	iPos—the PS position (1 to the maximum screen size) to copy the info to.
Example	This example copies a string to position 1761:
	iRc% = EHLLAPICopyStringToPS("Hello World", 1761)

EHLLAPISearchPS 3270 5250 VT

This function is used to search the entire presentation space (PS) for a given string.

Function	EHLLAPISearchPS (strString As String, iPos As Integer) As Integer
Input	StrString—the Visual Basic (VB) string to search.
Output	iPos—the PS position of the text, if found (if iRc = 0).
Example	The following example searches for the string "CP READ":

```
iRc% = EHLLAPISearchField( "More...", iPos% )
```

EHLLAPIDisconnect 3270 5250 VT

This function disconnects the Visual Basic (VB) interface from a session.

Function	EHLLAPIDisconnect (idSession As String) As Integer
Input	IdSession—the session short-name identifier. It must be a single, uppercase letter.
Example	The following example disconnects from session "A":

```
iRc% = EHLLAPIDisconnect( "A" )
```

EHLLAPISendFile 3270 5250 VT

This function is used to upload a file to the host. Common valid options are ASCII, CRLF, and APPEND. Other options are specific to the operating system. Separate all options with spaces, not commas.

Function	EHLLAPISendFile (strPCFileName As String, strSessionID As String, StrHostFilename As String, strOptions As String) As Integer
Input	StrPCFileName—contains the name of the computer file to upload. IdSession—the session short-name identifier. It must be a single, uppercase letter. StrHostFileName—contains the name of the HOST file to receive the data. StrOptions—contains the file-transfer options, separated by spaces.
Example	The following example transfers the computer file C:\AUTOEXEC.BAT to the host file PROFILE EXEC: iRc% = EHLLAPIReceiveFile("C:\AUTOEXEC.BAT", "A", "PROFILE EXEC", "ASCII CRLF")

EHLLAPIFindFieldPosition 3270 5250 VT

This function is used to get the presentation-space (PS) position of the previous or next protected or unprotected field. The search types are defined in the Declarations section of the HLLCALLS.BAS module.

Function	EHLLAPIFindFieldPosition (strSearchType As String, iPos As Integer) As Integer
Input	StrSearchType—the string-search type, which corresponds to the direction and field type. iPos—the PS position (1 to the maximum screen size) to begin the search.
Output	iPos—the PS position of the field.
Example	The following example searches for the next unprotected field starting at position 81: iPos% = 81 iRc% = EHLLAPIFindFieldPosition(EHLLAPI_NEXTUNPROT, iPos)

EHLLAPISendKey 3270 5250 VT

This function is used to send a sequence of keystrokes to the session. Although you can transfer data into the session using the CopyString functions, only this function allows you to press action keys such as Enter and PFxx.

Function	EHLLAPISendKey (strKeyString As String) As Integer
Input	StrKeyString—the VB string that contains the list of keys to press.
Example	The following example types the string "LOGIN PIERRE" and then presses the Enter key: iRc% = EHLLAPISendKey("LOGIN PIERRE@E")

Related Topics

Creating Special Key Strings

Creating Special Key Strings

The following entries assume that the ESC (escape) key is mapped to the '@' symbol.

To use a special key string, insert the @ symbol twice in the string by pressing the ESC key twice.

For example, type:

"PIERRE@@CC0" as "PIERRE@CC0", not "PIERRE", followed by the Clear key.

EHLLAPIGetRowString 3270 5250 VT

This function (which is not a true EHLLAPI call) is used to retrieve the contents of a specific row in the PS. Attributes are translated to blanks and no extended attributes are returned.

Function	EHLLAPIGetRowString (idSession As String, strString As String, iRow As Integer) As Integer
Input	IdSession—the session short-name identifier. It must be a single, uppercase letter. iRow—the row value between 1 and the maximum number of rows.
Output	StrString—the VB string to receive the contents of the row.
Example	The following example retrieves the contents of row 23 into a VB string: iRc% = EHLLAPIGetRowString("A", strRow$, 23)

EHLLAPISetCursorLocation 3270 5250 VT

This function is used to set the cursor position.

Function	EHLLAPISetCursorLocation (iNewPos As Integer) As Integer
Input	iPos—the presentation-space (PS) position of the cursor.
Example	The following example sets the cursor position:

```
iRc% = EHLLAPISetCursorLocation( 1761 )
```

EHLLAPIGetVersion 3270 5250 VT

This function is used to retrieve the version of the EHLLAPI interface.

Function	EHLLAPIGetVersion (hiVer As Integer, loVer As Integer) As Integer
Output	HiVer—the high-order digit of the version.
	LoVer—the low-order digit of the version.
Example	The following example retrieves the current version of the EHLLAPI interface:
	iRc% = EHLLAPIGetVersion(hiVer%, loVer%)

EHLLAPISetSessionParameters 3270 5250 VT

This function is used to set the current session parameters.

Function	EHLLAPISetSessionParameters (strParameters As String) As Integer
Input	StrParameters—the Visual Basic (VB) string containing the comma-separated options.
Example	The following example translates Extended Attributes (EABs) into CGA colors:

```
iRc% = EHLLAPISetSessionParameters( "XLATE,NOEAB" )
```

Related Topics

Valid Options

Valid Options 3270 5250 VT

You can use the following options when setting current session parameters:

ATTRB—Return attributes in Copy functions.

NOATTRB—Convert attributes to blanks.

EAB—Return extended attributes in functions. Although supported by EHLLAPI, the current VB functions currently disable this option.

NOEAB—Do not return extended attributes.

XLATE—Translate all unknown EBCDIC data to blanks.

NOXLATE—Do not translate any PS data.

AUTORESET—Unlock the keyboard before executing any EHLLAPISendKey function.

NORESET—Do not unlock the keyboard before executing EHLLAPISendKey.

SRCHALL—Search the entire PS space for strings.

SRCHFROM—Search from the specified position for strings.

SRCHFRWD—Search forward for a string.

SRCHBKWD—Search backward for a string.

FPAUSE—Insert a full pause for EHLLAPIPause.

IPAUSE—Insert an interruptible pause (if the PS is updated) for EHLLAPIPause.

QUIET—Perform Quiet mode file transfers.

NOQUIET—Perform a normal file transfer (a dialog box appears).

ESC=x—Set the EHLLAPISendKey escape character. The default value is @.

TIMEOUT=x—Specify the timeout for a file transfer, for example:

0—None

1—30 seconds

2—1 minute

3—1.5 minutes

4—2 minutes

5—2.5 minutes

6—3 minutes

7—3.5 minutes

8—4 minutes

9—4.5 minutes

J—5 minutes

K—5.5 minutes

L—6 minutes

M—6.5 minutes

N—7 minutes

EHLLAPIPause 3270 5250 VT

This function is used to pause for a specific number of half-second increments.

Function	EHLLAPIPause (iHalfSeconds As Integer) As Integer
Input	IHalfSeconds—the number of half-seconds to wait.
Example	The following example waits for 1.5 seconds:

```
iRc% = EHLLAPIPause( 3 )
```

EHLLAPIWait 3270 5250 VT

This function is used to wait for an event to occur on the host session. The function can check the status, wait for a certain amount of time, or wait indefinitely for the host PS to change. The wait types are defined in the Declarations section of the HLLCALLS.BAS module. Use the EHLLAPISetSessionParameters call to set the timeout used by the EHLLAPI_TWAIT option.

Function	EHLLAPIWait (iWaitType As Integer) As Integer
Input	iWaitType—the wait type as defined in the Declarations section of HLLCALLS.BAS.
Example	The following example checks if the PS updates: iRc% = EHLLAPIWait(EHLLAPI_NWAIT)
Valid Options	EHLLAPI_NWAIT—no wait. Check module status and return immediately. EHLLAPI_TWAIT—timed wait. Wait until the host updates or wait up to 60 seconds, whichever comes first. EHLLAPI_LWAIT—wait until the host system updates (forever).

EHLLAPIQueryCursorLocation 3270 5250 VT

This function returns the current cursor location.

Function	EHLLAPIQueryCursorLocation (iPos As Integer) As Integer
Output	iPos—the presentation-space (PS) position of the cursor.
Example	The following example retrieves the cursor position:

```
iRc% = EHLLAPIQueryCursorLocation( iPos% )
```

Visual Basic Interface

HostExplorer implements support for Visual Basic (VB) using a number of helper functions. These functions extend the EHLLAPI interface into VB, allowing you to communicate with the emulator and control it.

VB functions shield VB from improper data manipulation and prevent general protection faults.

Related Topics

Using the Visual Basic Interface

Visual Basic Return Codes

Using the Visual Basic Interface

To use the Visual Basic interface:

1. Include the HLLCALLS.BAS file in your project.

2. Before calling any EHLLAPI functions, call EHLLAPIQuerySessions. This determines which, if any, 3270 sessions are available. The returned string contains the short names of the available sessions.

All EHLLAPI functions return the EHLLAPI return code, as described in the EHLLAPI Programming Guide.

Visual Basic Return Codes

All EHLLAPI functions return the EHLLAPI return code, as described in the EHLLAPI Programming Guide. Before calling any EHLLAPI functions, call EHLLAPIQuerySessions. This determines which, if any, 3270 sessions are available. The returned string contains the short names of the available sessions.

Return codes:

0—Function completed successfully; PS is unlocked and ready for input.

1—Invalid PS position (null or blank with no connection).

2—File not sent. Command line is not valid or one or more unrecognized parameters; all recognized values accepted.

3—File transfer complete.

4—Successful connection, but PS is busy or timed out on TWAIT or LWAIT; or OIA copied, PS is busy.

5—Successful connection, but PS is locked or not all keystrokes could be sent or OIA copied, PS is locked.

6—Copy was completed, but data was truncated.

7—Invalid PS position.

8—No prior Function 23 or 50 call for this PS position.

9—System error, function failed. Emulator not loaded.

21—OIA was updated.

22—PS was updated.

23—OIA and/or PS was updated.

24—Search string was not found.

26—PS or OIA has been updated.

27—File transfer ended by user request.

28—Field length of 0 bytes.

WinHLLAPI Module

The WinHLLAPI module (WHLLAPI.DLL) fully implements Windows HLLAPI version 1.1 as defined in the Windows Open Services Architecture. The documentation (WHLLAPI.HLP) and development files (WHLLAPI.H, WHLLAPI.LIB, WHLLAP32.LIB) are installed on your system. This interface is available with both the 16-bit and 32-bit versions of HostExplorer.

Make sure that the appropriate DLL file (ACS3EHAP.DLL/EHLLAP32.DLL, HLLAPI.DLL, or WHLLAPI.DLL/WHLLAP32.DLL) is in your path by copying the appropriate DLL to your client application directory. This allows Windows to load the DLL when you run your client application.

Note: You may have to rename the 32-bit WHLLAP32.DLL to WHLAPI32.DLL to be compatible with the Attachmate® file name.

Window Close (201)

HostExplorer includes a special EHLLAPI/WinHLLAPI function to close a given window, regardless of any of the system flags. You can use Function 201 to close a specific short name window. The calling sequence is:

Function Number	201
Data String	A one-character presentation space name. Short name must be a letter of the alphabet (A–Z) or a number (1 to 5) for dynamic sessions.
Length	Not applicable.
PS Position	Not applicable.
Return	0—Window closed successfully. 1—Invalid PS short name specified. 9—No sessions active. Emulator not loaded.
Example	HLLFunc = 201; HLLDataString[0] = 'A'; HLLAPI(&HLLFunc, HLLDataString, &HLLDataLength, &PsPos);

Related Topics

Special EHLLAPI and WinHLLAPI Flags

EHLLAPI Support in VT and NVT mode

EHLLAPI Development files

NVT Mode Functions

NVT Mode Exceptions

ConnectPS (1)

HostExplorer supports a modified version of the ConnectPS function, which lets you dynamically create a new session to any host from EHLLAPI/WinHLLAPI. The calling sequence is:

Function Number 1

Data String Pointer to the following structure:

```
typedef struct _AUTOSTART_
{
char cPSID; // Sessions ID = '*' (Asterisk)
char cModelType; // Emulation Type = '3' for 3270, '5' for 5250, or 'V'

  for VT
char cReserved1;
char cReserved2;
char cReserved3;
char acHostName[128]; // Name/IP Address of host
} AUTOSTART, FAR *LPAUTOSTART;
```

Length Not applicable.

PS Position Not applicable.

Return

0—ConnectPS successful.

1—Invalid PS short name specified

9—No sessions active. Emulator not loaded.

Example

```
HLLFunc = 1;

HLLDataString[0] = '*';

HLLDataString[1] = '3';

strcpy( &HLLDataString[5], "1.2.3.4" );

HLLAPI(&HLLFunc, HLLDataString, &HLLDataLength, &PsPos );
```

Irma Compatibility Mode

By default, the EHLLAPI (ACS3EHAP.DLL) module is compatible with the Attachmate® Extra! for Windows specifications.

To enable Irma compatibility, add the following line to the EHLLAPI.Setting section in the HOSTEX.INI file:

[EHLLAPI.Settings]
Compatibility = *Irma*

Configuration Tips

HostExplorer enables you to associate a PS short name with a specific profile. This enables HLLAPI to spawn (start) a new session when you issue a ConnectPS command. Issuing a ConnectPS command lets you start your HLLAPI application without pre-loading the emulator. To associate a PS short name with a profile, select the Save Profile option from the File menu. The Save Profile dialog box lets you change the HLLAPI short name.

If you want the emulator to automatically assign valid HLLAPI letters to new sessions, add the following line to the System.Settings section in the global hostex.ini file, located in the HostEx directory where the user files are stored on your machine. For the appropriate directory path for your platform, refer to the list of the default locations for the user files. 

[System.Settings]
HLLAPI Auto Assign = On

In the event that HLLAPI automatically loads the emulator, it tries to spawn hostex32.exe. If this is not the program name or if the program name is not in the path, the Windows directory or Windows System directory specifies the program name.

To specify the program name, add the following line to the hostex.ini file in the HostEx directory where the user files are stored on your machine.

[EHLLAPI.Settings]
Auto Start Name = [path]programname.exe

Note: The user specifies the user directory when you install HostExplorer.

What is DDE?

You can use Dynamic Data Exchange (DDE) to carry out interprocess communication. It allows programs such as Excel, Word for Windows, and Visual Basic to interact with the 3270 emulator. The DDE interface enables you to create new terminal sessions, enter data, run macros, retrieve screens, and transfer files.

Unlike the EHLLAPI interface, a low-level programmatic interface that uses C or C++, the DDE interface in HostExplorer is designed to be used with high-level languages such as Visual Basic or Word Basic. For example, using DDE, you can write macros in Word for Windows that log you into the mainframe, transfer a file to the mainframe, then send the file automatically as e-mail.

The DDE interface included with HostExplorer is almost completely compatible with the Attachmate® EXTRA! for Windows DDE interface. This compatibility reduces the amount of work involved when you move applications to HostExplorer.

Related Topics

How Does DDE Work?

DDE Terminology

In DDE, the screen is called the presentation space (PS). When copying information to or from the PS, the indices used always begin at 1 and end at the last value of the PS. For example, a 24x80 screen has 1920 addressable positions, which range from 1 to 1920.

Some of the requests return multiple items back to you in a single string. Each item is delimited using the carriage return byte (“\r” in standard C syntax), value 0x0D, or 13 decimal.

DDE lets you start a conversation with:

· HostExplorer

· Word for Windows

· Microsoft Excel

Related Topics

DDE Samples

Starting a Conversation with HostExplorer

To start a conversation with HostExplorer, send an INITIATE message with HOSTEX as the application name and a topic that can be one of the session short names or System.

Starting a Conversation with Word for Windows

To start a conversation with Word for Windows, use the following syntax in a macro:

iChanNum = DDEInitiate("HOSTEX", "A")

You can replace the topic "A" with any existing session short name or "System" to access the system topic.

Starting a Conversation with Microsoft Excel

To start a conversation with Microsoft Excel, use the following syntax in a cell:

=INITIATE("HOSTEX", "A")

DDE Sample Code

The following is an example of logging in to a CMS account, written in Word for Windows Basic language. You can use the sample WordBasic macro to copy the current screen image from HostExplorer (configured as EHLLAPI session "A") to your Word document at the current insertion point. The macro uses DDE to connect to the emulator and copy the emulator line by line.

In the DDE sample code, HostExplorer lets you create login scripts to connect to remote hosts.

The first command is the On Error Command. Always include an On Error Command to make sure that the DDE link is terminated when the macro is finished. If you do not use the command, you may use all available DDE sources and be unable to execute the macro properly.

The macro begins by retrieving the number of rows and columns in the current presentation space. Because data returned by DDE is always in string format, you must use the Val() function to convert the data to numeric values. The macro proceeds to copy the screen line by line. The *request$* = ... line is where all the work is prepared.

This command builds a string of the format *PxxLyy*, where *xx* is the screen position (based from 1) and *yy* is the line length. Because of this, the macro issues requests for data such as *P1L80*, *P81L80*, and *P161L80*, to copy line by line. The data is then inserted into the current document using the Insert command. The last step of the macro is to close down the DDE connection.

```
Sub Main
 ChanNum = DDEInitiate( "HOSTEX", "A" )
 DDEPoke ChanNum, "Keystroke", "LOGIN PIERRE@E"
 DDEExecute ChanNum, "[Wait Unlock(6)]"
 DDEPoke ChanNum, "Keystroke", password@E
 DDETerminate ChanNum
 MessageBox "Logged into CMS successful", "Information"
End Sub
```

```
Sub Main
 On Error Goto ErrorHandler
 crlf$ = Chr$(13) + Chr$(10)
 iChanNum = DDEInitiate("HOSTEX", "A")
 iNumRows = Val(DDERequest$(iChanNum, "Rows"))
 iNumCols = Val(DDERequest$(iChanNum, "Columns"))
 If iChanNum Then
 For row = 1 To iNumRows
  request$ = "P" + Mid$(Str$(1 + ((row - 1) * iNumCols)), 2) + "L" + LTrim$(Str$(iNumCols))
 Data$ = Data$ + DDERequest$(iChanNum, request$) + crlf$
 Next row
 Insert Data$
 Else 'could not open session
  MsgBox "Could not open DDE Session with program."
 End If
 ErrorHandler:
 DDETerminate iChanNum
End Sub
```

Related Topics

DDE Terminology

DDE Terminology

In DDE, the screen is called the presentation space (PS). When copying information to or from the PS, the indices used always begin at 1 and end at the last value of the PS. For example, a 24x80 screen has 1920 addressable positions, which range from 1 to 1920.

Some of the requests return multiple items back to you in a single string. Each item is delimited using the carriage return byte (“\r” in standard C syntax), value 0x0D, or 13 decimal.

DDE lets you start a conversation with:

· HostExplorer

· Word for Windows

· Microsoft Excel

Related Topics

DDE Samples

How Does DDE Work?

DDE transfers information in conversations. A conversation occurs between a client application and a server application, such as HostExplorer.

When HostExplorer is loaded, it broadcasts to DDE that its services are available. Then a client application, such as Word for Windows, can initiate a conversation with HostExplorer. The procedure is similar to a telephone conversation: you must call a friend in order to have a conversation. Once the client and server applications begin a conversation, the client application can request information, run macros, press keys, and transfer files.

A client application can issue the following four types of DDE messages:

Advise

Execute

Poke

Request

To have a DDE conversation, you need the following fields:

Application Name

Topic

Related Topics

System Topic

Advise Message

You can use the Advise message to receive feedback about when certain events take place, such as updates to cursor movement or presentation space. In DDE terminology, these updates are known as warm links and hot links. They allow your application to receive updated information when a specified event takes place.

Hot links update your client information automatically whenever changes occur in HostExplorer, whereas warm links require additional steps to update your client information. Hot links are used to provide continuous, up-to-date information.

Example	The following function requests feedback when the cursor changes position.
Word for Windows	Insert a field in your document by pressing the INSERT FIELD key, then Ctrl+F9. Type the DDE command below for a warm or hot link. Do not type the curly brackets; they are simply the field-delimiter characters. { dde HOSTEX session name item name } { ddeauto HOSTEX session name item name }
Example	{ dde HOSTEX A Cursor}

Related Topics

Advise Commands

Advise Commands

You can send the following Advise commands to a session topic:

Alarm—Informs you that the terminal alarm has been sounded.

Cursor—Informs you that the cursor position has changed in the presentation space. HostExplorer returns the new cursor position in a hot link.

File Transfer— Informs you when a file transfer terminates in the presentation space. The first string returned is the PS short name followed by a “0”. This indicates that the file has been transferred.

OIA—Informs you that you have made changes to the operator information area (OIA) and returns the updated OIA string.

Power—Always returns "On".

PS—Informs you when the presentation space has changed and returns the complete PS as a string.

Related Topics

Execute Commands

Poke Commands

Request Commands

Execute Commands

You can issue the following commands to a session topic:

Allow Emulator Updates—Enables HostExplorer to update the 3270 window when it receives information from the host.

Block Emulator Updates—Prevents HostExplorer from updating its window when it receives information from the host.

End Session—Terminates the current terminal session. This command is identical to selecting Close Session from the File menu in HostExplorer.

Pause—Pauses for the specified time in half-second increments. The following example pauses for 2 seconds: [pause(4)]

Receive File—Downloads the specified file from the host to your computer. This requires command syntax. 

Send File—Uploads the specified file to the host from your computer. This requires command syntax. 

Run Macro—Runs the specified macro. The following example runs the macro dothis: [run macro(dothis)]

Wait Unlock—Pauses for the specified time in half-second increments until the 3270 keyboard is unlocked. If the keyboard is already unlocked, the function returns immediately. The following example pauses for up to three seconds: [Wait Unlock(6)]

Related Topics

Advise Commands

Poke Commands

Request Commands

Receive File command syntax

Command Syntax

The command syntax for TSO and MUSIC is:
[receive file(PC_filename host_filename transfer_options)]

The command syntax for CMS and CICS is:
[receive file(PC_filename host_filename (transfer_options))]

Examples

TSO or MUSIC
[receive file(c:\martin host.martin ASCII CRLF)]

CMS
[receive file(c:\vincent.txt vincent text (ASCII CRLF)]

CICS
[receive file(c:\test.txt test (ASCII CRLF))]

Send File command syntax

Command Syntax

The command syntax for TSO and MUSIC is:
[send file(PC_filename host_filename transfer_options)]

The command syntax for CMS and CICS is:
[send file(PC_filename host_filename (transfer_options))]

Examples

TSO or MUSIC
[send file(c:\martin host.martin ASCII CRLF)]

CMS:
[send file(c:\vincent.txt vincent text (ASCII CRLF)]

CICS
[send file(c:\test.txt test (ASCII CRLF))]

Poke Commands

You can send the following items to a Session topic:

Cursor—Sets the cursor position to the new value in the presentation space. The syntax for the value field is either nnn or Fnn[U/P]. nnn sets the position to the numeric value specified whereas Fnn[U/P] sets the position to the first position of the specified field. For additional information, see the Cursor example. 

EscChar—Sets the escape character used for sending keys. The default escape character is '@'.

Keystroke—Presses a collection of keystrokes. The string can contain up to 255 characters. The string format for keys is in HLLAPI mnemonic format. This allows you to enter normal text and press 3270 action keys such as Home, Pfx, and Clear. For additional information, see the Keystroke example. 

PS—Inserts the string into the entire presentation space. In this mode, data over protected fields is ignored. Therefore, you can retrieve the entire PS, update certain portions, then replace the entire PS.

Pnnnn[F/Lmmm]—Inserts the string into a specific position in the presentation space. Data is inserted until the end of the field or end of the data string, whichever comes first.

P100—Inserts the string at position 100 until the end of field.

P100F—Inserts the string in the field that contains position 100.

P100L20—Inserts the string at position 100 for a maximum length of 20 characters, regardless of the length of the string.

Fnn[U]—Inserts the string into a specific field in the presentation space. Data is inserted until the end of the field or end of the data string, whichever comes first.

F2—Inserts the string starting in the first position of the second field.

F2U—Inserts the string starting in the first position of the second unprotected field.

Rxx—Inserts the string into the specified row in the presentation space. Data is written only into unprotected fields.

R2—Inserts the string into the second row of the presentation space.

Search—Sets the search string for the Search request command.

Related Topics

Advise Commands

Execute Commands

Request Commands

Cursor Example

The following is a Cursor example:

"1"—sets the cursor to position 1.

"F2"—sets the cursor to the first position of the second field in the presentation space.

"F2U"—sets the cursor to the first position of the second unprotected field in the presentation space.

"F2P"—sets the cursor to the first position of the second protected field in the presentation space.

Keystroke Example

The following is a Keystroke example:

"LOGIN PIERRE@E"

Request Commands

You can request the following items from a Session topic:

Columns—Returns the number of columns in the current presentation space.

Cursor—Returns the current cursor location in the presentation space. 1 is the first position.

Emulator—Returns the window handle of the window displaying the presentation space.

File Transfer— Returns the short name of the presentation space and either 0 if no transfer is occurring or 1 if a transfer is occurring. For additional information, see the File Transfer example. 

Keyboard—Returns the status of the 3270 keyboard. Valid return values are Clear and Locked.

Model—Returns the 3270 model for the presentation space. Valid values are 2, 3, 4, and 5.

OIA—Returns the operator information area (OIA) in ASCII format.

Power—Always returns “On”.

Profile Name—Returns the name of the profile used for the presentation space; for example, DEFAULT.

Rows—Returns the number of rows in the current presentation space.

Search—Returns the position of the search string in the presentation space (PS). The search string is defined using the POKE command.

PS—Returns the entire contents of the PS. The entire space is returned as one string. Nulls are converted to blanks. Therefore, if you are using a Model 2 terminal (24x80), a string of 1920 bytes is returned.

Pnnnn[F/Lmmm]—Returns a portion of the PS. Nulls are converted to blanks. For additional information, refer to the Pnnnn[F/Lmmm] example. 

Fnn[U/P]—Returns the contents of the field specified. Nulls are converted to blanks. For additional information, refer to the Fnn example. 

Rnn—Returns the contents of the specified row. Nulls are converted to blanks.

R2—Returns the contents of the second row in the presentation space. The length is dependent on which 3270 model you are using. Models 2, 3, and 4 return 80 characters, whereas a model 5 returns 132 characters.

Related Topics

Advise Commands

Execute Commands

Poke Commands

File Transfer Example

The following is a File Transfer example:

A 1—Transfer occurring on session A.

Pnnnn[F/Lmmm] Examples

The following list consists of Pnnnn[F/Lmmm] examples:

P100—Returns the presentation space from position 100 to the end of the field containing position 100.

P100F—Returns the entire field that contains position 100.

P100L20—Returns data from position 100 for a length of 20 characters regardless of field positions.

Fnn Examples

The following list consists of Fnn examples:

F2— Returns the contents of the second field (unprotected or protected) in the presentation space.

F2U—Returns the contents of the second unprotected field in the presentation space.

F2P—Returns the contents of the second protected field in the presentation space.

Execute Message

You can use an Execute message to instruct HostExplorer how to perform commands, such as running macros and transferring files. An Execute message does not return any information.

Examples	The following function pauses the system for 1.5 seconds. The variable ChanNum represents the DDE conversation ID.
Word for Windows	DDEExecute ChanNum, "[pause(3)]"
Microsoft Excel	=EXECUTE(A1, "[pause(3))]")

Related Topics

Execute Commands

Poke Message

You can use a Poke message to send information to the DDE server HOSTEX. The Poke message lets you send information to identify a new cursor position, press keys, and set the search string. Poke does not return any information.

Examples	The following function sets the cursor position to position 1761. The variable ChanNum represents the DDE conversation ID.
Word for Windows	DDEPoke ChanNum, "Cursor", "1761"
Microsoft Excel 4.0	=POKE(A1, "Cursor", B1)
Microsoft Excel 5.0 and 7.0	You can poke only data that exists in a cell. DDEPoke ChanNum," Cursor ",Range("a1") If you want to Poke data in a text string, you must use the v4.0 macros.

Related Topics

Poke Commands

Request Message

You can use a Request message to retrieve information from the DDE server HOSTEX. The Request message lets you retrieve information such as the screen format, cursor position, and presentation space. Request messages retrieve information only. They do not perform any actions on the emulator or system.

Examples	The following function requests the third line of data from the presentation space. The variable ChanNum represents the DDE conversation ID.
Word for Windows	Data$ = DDERequest(ChanNum, "P160L80")
Microsoft Excel	=REQUEST(A1, "P160L80")

Related Topics

Request Commands

Application Name Field

The Application Name directs the DDE conversation to a particular application. The application name for HostExplorer is HOSTEX.

Topic Field

The Topic field specifies the terminal session, or session short name, assigned in the session profile. The terminal session is always a single letter and usually starts with A.

A special topic field called a SYSTEM topic field lets you perform functions that are not session-oriented. The Item field is a third and often used field. It usually specifies the request or command issued to the session or system.

System Topic

You can use the System Topic item to locate information about the system. For example, you can use the item to see which sessions are currently in use. To request any of the following, use the Request command.

The System Topic supports the following:

Formats—Returns the name of the DDE formats supported. Always returns "Text".

Profiles—Returns the list of defined profile names. Each item is separated by a carriage return character (0x0D).

Session Started—Returns the topic letter for the session last started with the Start Session EXECUTE command.

SysItems—Returns the list of system items that you can request. Each item is separated by a carriage return character (0x0D).

Topics—Returns the list of system topics that are currently available for conversations. Each item is separated by a carriage return character (0x0D).

Related Topics

System Topic commands

What is DDE?

System Topic Commands

The following is a list of available commands that you can issue to a SYSTEM topic:

Start Session—Starts a new session using the specified profile and connect options. For additional information, refer to the Start Session example. 

Profile Name—The profile name is always required along with its associated Profile Folder. For example:

[Start Session(VMTCP.3270_Profiles)]

[Start Session(VMTCP.a_folder,132.206.27.2)]

[Start Session(VMTCP.a_folder2,132.206.27.2,1023)]

Topic Name—The topic name for the session that just started can be retrieved from a System Request with the item "Session Started".

Related Topics

System Topic

What is DDE?

Start Session Example

The following is a Start Session example:

[Start Session(Profile_Name.Profile_Folder [,IP Host/Gateway Name [,IP Port]])]

Introducing FTP API

The FTP API is a non-OLE interface that lets you build Hummingbird Basic scripts to perform local and remote disk and directory operations. The FTP API functions correspond to functionality in the 5.3 and earlier versions of FTP. For information on the 6.0 version of FTP API, see Introducing FTP OLE API. If you require more information on the function of a specific FTP API scripting command, refer to the FTP online Help.

Note: You should also familiarize yourself with the following reference manuals for HLLAPI from IBM:

· IBM 3270 Personal Computer High Level Language Application Programming Interface

· AIX Version 3.2 High Level Application Programming Interface (HLLAPI) Programming

FTP script requirements

When creating an FTP script, you must fulfill certain requirements:

· All of the FTP API functions are prototyped in the ftprtns.ebh header file. In order to access the FTP API methods in Hummingbird DLL's, you must include the ftprtns.ebh header file (.ebh) in your script file.

· All FTP scripts must begin with the InitFTP command and end with the DeinitFTP command.

Handling FTP scripting errors

The GetErrorText command can be used to handle errors. All the error codes generated by FTP API scripting commands are numbered between 3000 and 4000.

Account function

This function sends an FTP account command to the host. In order to transfer files, you must supply the account command to some hosts.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call Account(BYVAL account$)
Parameters	account—specifies the account string you want to send to the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

AppendFile function

This function appends a single file from your PC to the host. To specify the destination as the current directory on the host, type an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call AppendFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	sourcePath—specifies the full source path of the PC file name you want to send to the remote host. destPath—specifies the full destination path of the host file name to which you want to append the file.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

Sample client source code

The source code for sample scripts is included in the EB subdirectory of your common applications *home* directory.

These samples demonstrate the Syntax and usage of Hummingbird Basic Language commands and API commands. You may find it useful to refer to these script files for help when you are creating your own script files. To open the source file (.ebs) in Hummingbird Basic Workbench, click Open on the File menu.

To print or run the loaded source file click Print or Run on the File menu.

Sample Scripts

TestFTP.ebs—is a sample script that demonstrates the usage of Hummingbird File Transfer Protocol Application Programming Interface.

Note: The X client sample scripts are available only if you purchased Exceed.

ASCIIType setting

This setting changes the current file transfer type to ASCII format. The default transfer type for FTP scripts is AUTO.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call ASCIIType()
Parameters	This method has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

BinaryType setting

This setting changes the current file transfer type to binary format. The default transfer type for FTP scripts is AUTO.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call BinaryType()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

ChangeHostDir function

This function changes the current directory to the specified directory. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call ChangeHostDir(BYVAL dirName$)
Parameters	dirName—specifies the directory name or full path to which to change the Host Directory.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

ConnectToHost function

This function connects to a remote host. You must supply the server port in order for this command to make the connection. The host name can either be an IP address or a server name specified in the host table.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call ConnectToHost(BYVAL host$, BYVAL serverPort as portint)
Parameters	host—specifies the remote host string name or IP address to which you want to connect. serverPort—specifies the port you want to use to make the connection.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

CreateValidFATFiles setting

This function sets a flag which the FTP script uses during PC file creation to create valid FAT file names. If the parameter is nonzero the flag is set, otherwise it is reset. By default this flag is set and the FTP script creates valid FAT file names. The CreateValidFATFiles method returns a nonzero value upon a successful setting, or zero to indicate an error.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	returnVal=CreateVAlidFATFiles(BYVAL setVal as portint)
Parameters	setVal—specifies an integer that turns valid file creation on the PC on or off.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

CTRLZStatus setting

This function determines whether or not the CTRL-Z character is being inserted at the end of ASCII files, when they are transferred from the PC to the server. A nonzero value indicates that the FTP session will insert a CTRL-Z character at the end of transferred files. Otherwise, a returned zero indicates that CTRL-Z is not inserted at the end of transferred files.

BASIC Syntax	returnVal=CTRLZStatus()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DeinitFTP function

This function de-initializes the environment of the FTP-specific environment variables, and must be the last method executed in the Basic main method.

BASIC Syntax	call DeinitFTP()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DeleteHostFile function

This function deletes the specified file from the host. The argument may include any wildcard characters supported by the host.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call DeleteHostFile(BYVAL sourcePath$)
Parameters	SourcePath—specifies the path of the file to be deleted from the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DelTree function

This function deletes an entire tree from the host. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call DelTree(BYVAL dirName$)
Parameters	dirName—specifies the directory name and its subdirectories that you will delete

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DisconnectFromHost function

This function disconnects the current connection. It generates an error if there are no established connections. You can retrieve the text associated with the function using the error-handling routine.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call DisconnectFromHost()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

FTPActive setting

This setting forces the FTP client session to establish the data connection actively. This makes the server (host) passive. By default, FTP scripting sessions are active.

BASIC Syntax	call FTPActive()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

FTPDIR function

This function gets the directory listing of the specified directory with or without options in the argument. The returned text will include a full description of each file on the host. If required, you can further process the returned text using the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = FTPDIR(BYVAL options$)
Parameters	options—specifies the full path, or options that may be supplied to the host (this varies with each host).

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

FTPPassive setting

This setting forces the FTP session to establish the data connection passively. This makes the server (host) active and the FTP client passive. By default, FTP scripting sessions are active.

BASIC Syntax	call FTPPassive()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetErrorText function

This function retrieves the error text associated with the last FTP subroutine or function call. The FTP scripting engine generates the text.

BASIC Syntax	Text$ = GetErrorText(BYVAL errorCode as portint)
Parameters	ErrorCode—specifies the trapped error code generated as a result of the previous FTP functions or subroutines.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetFile function

This function gets a single file from the host to the PC. To specify the destination as the current directory on the PC, pass an empty string as the destination path. If the current host directory does not include the source file, you should specify the full path. GetFile does not accept wildcard specifications.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call GetFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	*source path*—specifies the full path to the file on the host. *Destination path*—specifies the full path to where you want to put the file on the PC.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetLocalFilenamesStatus setting

This setting returns the current state of the flag which FTP scripting uses to transfer local files with lowercase file names during a PUT operation. The function will either return a nonzero value indicating that the flag is set, or a zero value to indicate that the flag is not set.

BASIC Syntax	returnVal = GetLocalFilenamesStatus()
Parameters	This setting has no parameters.

GetOutputText function

This function retrieves the replies made by the server with respect to any other FTP API function.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = GetOutputText
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

The GetOutputText function is used to retrieve the replies made by the server to any used subroutines or functions. It returns the string echoed by the server when a command is sent, allowing you to see what the server has returned.

GetReplyTimeoutValue setting

This setting returns the current timeout value used by the FTP scripting tool. The value returned represents how long the FTP scripting tool waits for a reply from the remote host after an FTP command is sent. The GetReplyTimeoutValue function returns a long value indicating the timeout.

BASIC Syntax	ReturnVal = GetReplyTimeoutValue()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetTransferType setting

This function returns the current transfer type. One of the following values will be returned: ASCII, BINARY or AUTO DETECT. In AUTO DETECT transfer type reverts to ASCII when transferring text files, and to BINARY when transferring binary files (e.g. executables).

The default transfer type for FTP scripts is AUTO.

BASIC Syntax	ReturnVal = GetTransferType()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetTree function

This function gets an entire tree from the host to the PC. To specify the current directory as the destination on the PC, pass an empty string as the destination path. If the current host directory does not include the source directory, you should specify the full path. GetTree does not accept wildcard specifications.

BASIC Syntax	call GetTree(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the full source path to be received from the host. destPath—specifies the full destination path on the PC.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

InitFTP function

This function initializes the environment for the FTP scripting functions. This function must be called before you make any FTP scripting request. Otherwise, the scripting subroutines behave abnormally. A nonzero integer return value specifies a successful initialization, otherwise a zero value is returned.

BASIC Syntax	ReturnVal = InitFTP()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

LocalDelTree function

This function deletes an entire tree from the local PC. The argument must not be empty. LocalDelTree is a supplement to the current Hummingbird Scripting Language API. No connection to any host is required, however, you must use the FTPInit and Deinit API.

Errors should be trapped by the provided error-handling function.

BASIC Syntax	call LocalDelTree(BYVAL dirName$)
Parameters	dirName—specifies the path of the tree to be deleted from the PC.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

LS function

This function gets the directory listing of the specified directory, with or without options in the argument. If required, you can process the returned text further using the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = LS(BYVAL options$)
Parameters	options—specifies the full path, or options that may be supplied to the host (this varies with each host).

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

MakeHostDir function

This function makes a new directory on the host. The argument must not be empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call MakeHostDir(BYVAL dirName$)
Parameters	dirName—specifies the new directory name or full path.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

MGet function

This function gets multiple files from the host to the current PC directory.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call Mget(BYVAL pattern$)
Parameters	pattern—specifies the pattern used by the host to get all matching file. The pattern may include wildcards native to the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

MPut function

This function puts multiple files to the current directory on the host from the PC.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call Mput(BYVAL pattern$)
Parameters	pattern—specifies the pattern used by the PC to get all matching file. The pattern may include wildcards native to the local operating system.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

PrintWorkingDir function

This function acquires the full path of the current directory on the host. The returned string may include FTP command codes, and can be processed further, if required.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = PrintWorkingHostDir
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

PutFile function

This function moves a single file from the PC to the host. To specify the destination as the current directory on the host, pass an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call PutFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the full source path of the PC file name to be sent to the remote host. destPath—specifies the full destination path of the host file name.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

PutTree function

This function puts an entire tree from the PC to the host. To specify the destination as the current directory on the host, pass an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call PutTree(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the full source path of the PC path to be sent to the remote host. destPath—specifies the full destination path on the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

QuoteCommand function

This function sends arbitrary FTP commands to the host. The returned string may include FTP command codes, and, if required, can be processed further.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = QuoteCommand(BYVAL commandText$)
Parameters	CommandText—specifies the command text to be sent to the host

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

RemoveHostDir function

This function removes the specified directory from the host. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call RemoveHostDir(BYVAL dirName$)
Parameters	dirName—specifies the directory name or full path to be removed.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

RenameHostFile function

This function renames the specified file to a new name. The subroutine does not attempt to rename files unless you specify both arguments.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call RenameHostFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the path of the file to be renamed.
	destPath—specifies the path of the file with the new name.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

SessionType setting

This setting determines the active/passive status of the FTP client session. If the function returns a nonzero value, then the FTP client session is active and the remote host (server) is passive. Otherwise, a zero value indicates that the remote host (server) is active and the FTP client session is passive.

BASIC Syntax	ReturnVal=SessionType()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

SetReplyTimeoutValue setting

This setting sets how many seconds FTP scripting waits for a reply from the remote host after a command is sent. If the parameter is a positive, nonzero, long integer, the timeout value will be set. Otherwise, the function returns a zero, which indicates an error. By default the timeout value is set to 120 (secs).

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	ReturnVal = SetReplyTimeoutValue(BYVAL value as long)
Parameters	value—specifies a long value to be set as the timeout.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

SetLocalFilenamesLowercase setting

This setting sets a flag that FTP scripting uses during a put operation in order to transfer local files to the host with lowercase file names. If the parameter is nonzero, the flag sets. Otherwise, it resets. By default this flag is set, and FTP scripting will transfer lowercase file names.

BASIC Syntax	ReturnVal = SetTransferType(BYVAL transferType as portint)
Parameters	Transfertype—specifies an integer that turns lowercase transfer on or off.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

SetTransferType setting

This setting sets the current transfer type to ASCII, BINARY or AUTO DETECT type. If the function is successful, the previous transfer type is returned; otherwise, a negative value returns. In AUTO DETECT, transfer type reverts to ASCII when transferring text files, and to BINARY when transferring binary files (e.g. executables). The default transfer type for FTP scripts is AUTO.

BASIC Syntax	ReturnVal = SetTransferType(BYVAL transferType as portint)
Parameters	TransferType—specifies a valid integer value to be set as transfer type.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

StripCtrlZ setting

This setting removes the CTRL-Z if it is at the end of ASCII files being transferred from the PC to the server. This ensures that CTRL-Z does not exist in the server's file. When this flag is set, hclftp.dll ignores the state of the WriteCtrlZ flag. By default, CTRL-Z is inserted at the end of files.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call StripCtrlZ(BYVAL setVal as portint)
Parameters	setVal—specifies the Boolean variable used to set or reset the option..

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

SystemType function

This function requests the host to specify the underlying operating system of the host. The subroutine sends an FTP (SYST) command to the host. The server either returns an error specifying that the command has not been implemented, or returns the text that includes the server system type. This text can be retrieved with the GetOutputText function, and can be processed further by the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call SystemType()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

UserLogin function

This function logs you on to the remote server. If your server requires a password or account, you should type them as specified below; otherwise, leave these strings empty. This function generates an error if there are no established connections.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call UserLogin(BYVAL userName$, BYVAL password$, BYVAL account$)
Parameters	userName—specifies the username that should be used to log on to the remote host. password—specifies the password to be used for login. account—specifies the account to be used for login.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

ValidFATFileCreate setting

This setting returns the current state of the flag that FTP scripting uses during PC file creation to create valid FAT file names. The function either returns a nonzero value indicating that the flag is set, or a zero value indicating that the flag is not set

BASIC Syntax	ReturnVal = ValidFATFileCreate()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

WriteCtrlZ setting

This setting controls whether or not a CTRL-Z end-of-file marker is inserted at the end of ASCII files being transferred from the PC to the server. By default, CTRL-Z is inserted at the end of files.

BASIC Syntax	call WriteCtrlZ(BYVAL setVal as portint)
Parameters	setVal—specifies the Boolean variable used to set or reset the option.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

Introducing FTP OLE API

The FTP API is a OLE interface that lets you build Hummingbird Basic scripts to perform local disk and directory operations. The FTP API methods and properties correspond to functionality in the current version of FTP. For information on FTP API for versions 5.2 or earlier, see Introducing FTP API. If you require more information on the function of a specific FTP API scripting command, refer to the FTP online Help.

Note: You should also familiarize yourself with the following reference manuals for HLLAPI from IBM:

· IBM 3270 Personal Computer High Level Language Application Programming Interface

· AIX Version 3.2 High Level Application Programming Interface (HLLAPI) Programming .

FTP OLE Objects

The FTP OLE API includes three objects:

· IHclFtpEngine

· IHclFtpSession

· IHclFtpSessions

These objects implement the standard IDispatch interface, and include a group of user-defined types common to all three objects.

All methods generate an error if there are no established connections. The text associated with the error can be retrieved with the error-handling routine. All errors that occur on the host side are generated, and must be trapped with the error-handling routine.

Introducing FTP API

The FTP API is a non-OLE interface that lets you build Hummingbird Basic scripts to perform local and remote disk and directory operations. The FTP API functions correspond to functionality in the 5.3 and earlier versions of FTP. For information on the 6.0 version of FTP API, see Introducing FTP OLE API. If you require more information on the function of a specific FTP API scripting command, refer to the FTP online Help.

Note: You should also familiarize yourself with the following reference manuals for HLLAPI from IBM:

· IBM 3270 Personal Computer High Level Language Application Programming Interface

· AIX Version 3.2 High Level Application Programming Interface (HLLAPI) Programming

FTP script requirements

When creating an FTP script, you must fulfill certain requirements:

· All of the FTP API functions are prototyped in the ftprtns.ebh header file. In order to access the FTP API methods in Hummingbird DLL's, you must include the ftprtns.ebh header file (.ebh) in your script file.

· All FTP scripts must begin with the InitFTP command and end with the DeinitFTP command.

Handling FTP scripting errors

The GetErrorText command can be used to handle errors. All the error codes generated by FTP API scripting commands are numbered between 3000 and 4000.

Account function

This function sends an FTP account command to the host. In order to transfer files, you must supply the account command to some hosts.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call Account(BYVAL account$)
Parameters	account—specifies the account string you want to send to the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

AppendFile function

This function appends a single file from your PC to the host. To specify the destination as the current directory on the host, type an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call AppendFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	sourcePath—specifies the full source path of the PC file name you want to send to the remote host. destPath—specifies the full destination path of the host file name to which you want to append the file.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

ASCIIType setting

This setting changes the current file transfer type to ASCII format. The default transfer type for FTP scripts is AUTO.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call ASCIIType()
Parameters	This method has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

BinaryType setting

This setting changes the current file transfer type to binary format. The default transfer type for FTP scripts is AUTO.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call BinaryType()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

ChangeHostDir function

This function changes the current directory to the specified directory. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call ChangeHostDir(BYVAL dirName$)
Parameters	dirName—specifies the directory name or full path to which to change the Host Directory.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

ConnectToHost function

This function connects to a remote host. You must supply the server port in order for this command to make the connection. The host name can either be an IP address or a server name specified in the host table.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call ConnectToHost(BYVAL host$, BYVAL serverPort as portint)
Parameters	host—specifies the remote host string name or IP address to which you want to connect. serverPort—specifies the port you want to use to make the connection.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

CreateValidFATFiles setting

This function sets a flag which the FTP script uses during PC file creation to create valid FAT file names. If the parameter is nonzero the flag is set, otherwise it is reset. By default this flag is set and the FTP script creates valid FAT file names. The CreateValidFATFiles method returns a nonzero value upon a successful setting, or zero to indicate an error.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	returnVal=CreateVAlidFATFiles(BYVAL setVal as portint)
Parameters	setVal—specifies an integer that turns valid file creation on the PC on or off.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

CTRLZStatus setting

This function determines whether or not the CTRL-Z character is being inserted at the end of ASCII files, when they are transferred from the PC to the server. A nonzero value indicates that the FTP session will insert a CTRL-Z character at the end of transferred files. Otherwise, a returned zero indicates that CTRL-Z is not inserted at the end of transferred files.

BASIC Syntax	returnVal=CTRLZStatus()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DeinitFTP function

This function de-initializes the environment of the FTP-specific environment variables, and must be the last method executed in the Basic main method.

BASIC Syntax	call DeinitFTP()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DeleteHostFile function

This function deletes the specified file from the host. The argument may include any wildcard characters supported by the host.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call DeleteHostFile(BYVAL sourcePath$)
Parameters	SourcePath—specifies the path of the file to be deleted from the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DelTree function

This function deletes an entire tree from the host. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call DelTree(BYVAL dirName$)
Parameters	dirName—specifies the directory name and its subdirectories that you will delete

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

DisconnectFromHost function

This function disconnects the current connection. It generates an error if there are no established connections. You can retrieve the text associated with the function using the error-handling routine.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call DisconnectFromHost()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

FTPActive setting

This setting forces the FTP client session to establish the data connection actively. This makes the server (host) passive. By default, FTP scripting sessions are active.

BASIC Syntax	call FTPActive()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

FTPDIR function

This function gets the directory listing of the specified directory with or without options in the argument. The returned text will include a full description of each file on the host. If required, you can further process the returned text using the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = FTPDIR(BYVAL options$)
Parameters	options—specifies the full path, or options that may be supplied to the host (this varies with each host).

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

FTPPassive setting

This setting forces the FTP session to establish the data connection passively. This makes the server (host) active and the FTP client passive. By default, FTP scripting sessions are active.

BASIC Syntax	call FTPPassive()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetErrorText function

This function retrieves the error text associated with the last FTP subroutine or function call. The FTP scripting engine generates the text.

BASIC Syntax	Text$ = GetErrorText(BYVAL errorCode as portint)
Parameters	ErrorCode—specifies the trapped error code generated as a result of the previous FTP functions or subroutines.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetFile function

This function gets a single file from the host to the PC. To specify the destination as the current directory on the PC, pass an empty string as the destination path. If the current host directory does not include the source file, you should specify the full path. GetFile does not accept wildcard specifications.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call GetFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	*source path*—specifies the full path to the file on the host. *Destination path*—specifies the full path to where you want to put the file on the PC.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetLocalFilenamesStatus setting

This setting returns the current state of the flag which FTP scripting uses to transfer local files with lowercase file names during a PUT operation. The function will either return a nonzero value indicating that the flag is set, or a zero value to indicate that the flag is not set.

BASIC Syntax	returnVal = GetLocalFilenamesStatus()
Parameters	This setting has no parameters.

GetOutputText function

This function retrieves the replies made by the server with respect to any other FTP API function.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = GetOutputText
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

The GetOutputText function is used to retrieve the replies made by the server to any used subroutines or functions. It returns the string echoed by the server when a command is sent, allowing you to see what the server has returned.

GetReplyTimeoutValue setting

This setting returns the current timeout value used by the FTP scripting tool. The value returned represents how long the FTP scripting tool waits for a reply from the remote host after an FTP command is sent. The GetReplyTimeoutValue function returns a long value indicating the timeout.

BASIC Syntax	ReturnVal = GetReplyTimeoutValue()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetTransferType setting

This function returns the current transfer type. One of the following values will be returned: ASCII, BINARY or AUTO DETECT. In AUTO DETECT transfer type reverts to ASCII when transferring text files, and to BINARY when transferring binary files (e.g. executables).

The default transfer type for FTP scripts is AUTO.

BASIC Syntax	ReturnVal = GetTransferType()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

GetTree function

This function gets an entire tree from the host to the PC. To specify the current directory as the destination on the PC, pass an empty string as the destination path. If the current host directory does not include the source directory, you should specify the full path. GetTree does not accept wildcard specifications.

BASIC Syntax	call GetTree(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the full source path to be received from the host. destPath—specifies the full destination path on the PC.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

InitFTP function

This function initializes the environment for the FTP scripting functions. This function must be called before you make any FTP scripting request. Otherwise, the scripting subroutines behave abnormally. A nonzero integer return value specifies a successful initialization, otherwise a zero value is returned.

BASIC Syntax	ReturnVal = InitFTP()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

LocalDelTree function

This function deletes an entire tree from the local PC. The argument must not be empty. LocalDelTree is a supplement to the current Hummingbird Scripting Language API. No connection to any host is required, however, you must use the FTPInit and Deinit API.

Errors should be trapped by the provided error-handling function.

BASIC Syntax	call LocalDelTree(BYVAL dirName$)
Parameters	dirName—specifies the path of the tree to be deleted from the PC.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

LS function

This function gets the directory listing of the specified directory, with or without options in the argument. If required, you can process the returned text further using the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = LS(BYVAL options$)
Parameters	options—specifies the full path, or options that may be supplied to the host (this varies with each host).

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

MakeHostDir function

This function makes a new directory on the host. The argument must not be empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call MakeHostDir(BYVAL dirName$)
Parameters	dirName—specifies the new directory name or full path.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

MGet function

This function gets multiple files from the host to the current PC directory.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call Mget(BYVAL pattern$)
Parameters	pattern—specifies the pattern used by the host to get all matching file. The pattern may include wildcards native to the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

MPut function

This function puts multiple files to the current directory on the host from the PC.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call Mput(BYVAL pattern$)
Parameters	pattern—specifies the pattern used by the PC to get all matching file. The pattern may include wildcards native to the local operating system.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

PrintWorkingDir function

This function acquires the full path of the current directory on the host. The returned string may include FTP command codes, and can be processed further, if required.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = PrintWorkingHostDir
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

PutFile function

This function moves a single file from the PC to the host. To specify the destination as the current directory on the host, pass an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call PutFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the full source path of the PC file name to be sent to the remote host. destPath—specifies the full destination path of the host file name.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

PutTree function

This function puts an entire tree from the PC to the host. To specify the destination as the current directory on the host, pass an empty string as the destination path.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call PutTree(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the full source path of the PC path to be sent to the remote host. destPath—specifies the full destination path on the host.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

QuoteCommand function

This function sends arbitrary FTP commands to the host. The returned string may include FTP command codes, and, if required, can be processed further.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	Text$ = QuoteCommand(BYVAL commandText$)
Parameters	CommandText—specifies the command text to be sent to the host

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

RemoveHostDir function

This function removes the specified directory from the host. You cannot leave the argument empty.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call RemoveHostDir(BYVAL dirName$)
Parameters	dirName—specifies the directory name or full path to be removed.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

RenameHostFile function

This function renames the specified file to a new name. The subroutine does not attempt to rename files unless you specify both arguments.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call RenameHostFile(BYVAL sourcePath$, BYVAL destPath$)
Parameters	SourcePath—specifies the path of the file to be renamed.
	destPath—specifies the path of the file with the new name.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

SessionType setting

This setting determines the active/passive status of the FTP client session. If the function returns a nonzero value, then the FTP client session is active and the remote host (server) is passive. Otherwise, a zero value indicates that the remote host (server) is active and the FTP client session is passive.

BASIC Syntax	ReturnVal=SessionType()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

SetReplyTimeoutValue setting

This setting sets how many seconds FTP scripting waits for a reply from the remote host after a command is sent. If the parameter is a positive, nonzero, long integer, the timeout value will be set. Otherwise, the function returns a zero, which indicates an error. By default the timeout value is set to 120 (secs).

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	ReturnVal = SetReplyTimeoutValue(BYVAL value as long)
Parameters	value—specifies a long value to be set as the timeout.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

SetLocalFilenamesLowercase setting

This setting sets a flag that FTP scripting uses during a put operation in order to transfer local files to the host with lowercase file names. If the parameter is nonzero, the flag sets. Otherwise, it resets. By default this flag is set, and FTP scripting will transfer lowercase file names.

BASIC Syntax	ReturnVal = SetTransferType(BYVAL transferType as portint)
Parameters	Transfertype—specifies an integer that turns lowercase transfer on or off.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

SetTransferType setting

This setting sets the current transfer type to ASCII, BINARY or AUTO DETECT type. If the function is successful, the previous transfer type is returned; otherwise, a negative value returns. In AUTO DETECT, transfer type reverts to ASCII when transferring text files, and to BINARY when transferring binary files (e.g. executables). The default transfer type for FTP scripts is AUTO.

BASIC Syntax	ReturnVal = SetTransferType(BYVAL transferType as portint)
Parameters	TransferType—specifies a valid integer value to be set as transfer type.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

StripCtrlZ setting

This setting removes the CTRL-Z if it is at the end of ASCII files being transferred from the PC to the server. This ensures that CTRL-Z does not exist in the server's file. When this flag is set, hclftp.dll ignores the state of the WriteCtrlZ flag. By default, CTRL-Z is inserted at the end of files.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call StripCtrlZ(BYVAL setVal as portint)
Parameters	setVal—specifies the Boolean variable used to set or reset the option..

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

SystemType function

This function requests the host to specify the underlying operating system of the host. The subroutine sends an FTP (SYST) command to the host. The server either returns an error specifying that the command has not been implemented, or returns the text that includes the server system type. This text can be retrieved with the GetOutputText function, and can be processed further by the script.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call SystemType()
Parameters	This function has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

UserLogin function

This function logs you on to the remote server. If your server requires a password or account, you should type them as specified below; otherwise, leave these strings empty. This function generates an error if there are no established connections.

Errors generated vary with each host, but the errors should be trapped by the provided error-handling function.

BASIC Syntax	call UserLogin(BYVAL userName$, BYVAL password$, BYVAL account$)
Parameters	userName—specifies the username that should be used to log on to the remote host. password—specifies the password to be used for login. account—specifies the account to be used for login.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

ValidFATFileCreate setting

This setting returns the current state of the flag that FTP scripting uses during PC file creation to create valid FAT file names. The function either returns a nonzero value indicating that the flag is set, or a zero value indicating that the flag is not set

BASIC Syntax	ReturnVal = ValidFATFileCreate()
Parameters	This setting has no parameters.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script.

WriteCtrlZ setting

This setting controls whether or not a CTRL-Z end-of-file marker is inserted at the end of ASCII files being transferred from the PC to the server. By default, CTRL-Z is inserted at the end of files.

BASIC Syntax	call WriteCtrlZ(BYVAL setVal as portint)
Parameters	setVal—specifies the Boolean variable used to set or reset the option.

Note: This function is prototyped in the ftprtns.ebh header file. In order to access the prototype for this API command, you must include this header file in your script file.

IHclFtpEngine Object

IHclFtpEngine Object provides information on the current application such as its name, parent and the state of the main window. Create IHclFtpEngine Object before you create any other object.

The methods and properties of this object are for use with this object and for version 6.0 or later only.

Related Topics

IHclFtpEngine Object properties

IHclFtpEngine Object methods

IHclFtpEngine Object properties

Application property

FullName property

Name property (IHclFtpEngine)

Parent property

Sessions property

Visible property

Related Topics

IHclFtpEngine Object method

Application property

This property returns the application object of IHclFtpEngine type.

BASIC Syntax	*variable* = Application
Parameters	This property has no parameters.
Data Type	IhclFtpSession
Property Type	Read Only

FullName property

This property returns the full name of the application.

BASIC Syntax	*variable* = FullName
Parameters	This property has no parameters.
Data Type	String
Property Type	Read Only

Name property (IHclFtpEngin)

This property returns the name of the application.

BASIC Syntax	*variable* = Name
Parameters	This property has no parameters.
Data Type	String
Property Type	Read Only

Parent property

This property returns the application object of IhclFtpEngine type.

BASIC Syntax	*variable* = Parent
Parameters	This property has no parameters.
Data Type	IhclFtpEngine
Property Type	Read Only

Sessions property

This is the most useful of the IHclFtpEngine properties. Returns an object of IHclFtpSessions type. This is the collection of IHclFtpSession objects.

BASIC Syntax	*variable* = Sessions
Parameters	This property has no parameters.
Data Type	IhclFtpSessions
Property Type	Read Only

Visible property

This property returns the state of the main window. If the window is visible, then the boolean is TRUE; if the window is not visible, then the boolean is FALSE.

BASIC Syntax	*variable* = Visible
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read Only

IHclFtpEngine Object method

Quit method

Related Topics

IHclFtpEngine Object properties

Quit method

This method exits the application.

BASIC Syntax	Quit
Parameters	This property has no parameters.
Property Type	None

IHclFtpSession Object

IHclFtpSession Object includes methods for executing a wide range of commands for managing files, printing, and connecting to the host. IHclFtpSession properties provide information on accounts, firewalls, servers, and passwords.

The methods and properties of this object are for use with this object and for version 6.0 or later only.

Related Topics

IHclFtpSession Object methods

IHclFtpSession Object properties

IHclFtpSession Object methods

AppendFile method
ChangeHostDir method
ConnectToHost method
DeleteHostFile method
DelTree method
DisconnectFromHost method
FTPDIR method
GetFile method
GetTree method
HostToHostFileAppend method
HostToHostFileTransfer method
HostToHostMultipleFileTransfer method
HostToHostTreeTransfer method
LS method
MakeHostDir method
MGet method
MPut method
NewUserLogIn method
ParentDir method
PrintHostFile method
PutFile method
PutTree method
QuoteCommand method
RemoteHelp method
RemoveHostDir method
RenameHostFile method
ReplyText method
SendAccountCommand method
ServerBanner method
SystemType method
UserLogIn method
WorkingHostDirectory method

Related Topics

IHclFtpSession Object properties

AppendFile method

This method appends a PC file to a host file. If the destination path is empty, the script sends the file to the current host working directory.

BASIC Syntax	AppendFile [*source path, destination path*]
Parameters	*source Path*—specifies the full source path of the PC file name you want to append to the remote host file. (String) *destination Path*—specifies the full destination path of the host file to which you want to append the PC file. (String)
Return Value	None

MPut method

This method sends multiple files from the PC to the host working directory.

BASIC Syntax	MPut [*pattern*]
Parameters	pattern—specifies the pattern used by the PC to get all matching files. The pattern may include wildcards native to the PC. (String)
Return Value	None

ChangeHostDir method

This method changes to another directory on the host.

BASIC Syntax	ChangeHostDir [*directory path*]
Parameters	*Directory path*—specifies the path of the directory to which you want to change. (String)
Return Value	None

NewUserLogin method

This method uses the parameters to authenticate another login with the new user name. All required parameters must be included.

BASIC Syntax	NewUserLogIn [*username*, *password*, *account*, *initialhost directory*]
Parameters	*userName*—specifies the username that should be used to log on to the remote host. *password*—specifies the password to be used for login. *account*—specifies the account to be used for login. *Initialhost directory*— specifies the directory to which the initial host directory is changed after the connection is made to the server at login
Return Value	None

ConnectToHost method

This method connects to the host. It uses the values of the properties.

BASIC Syntax	N/A
Parameters	This method has no parameters.
Return Value	None

ParentDir method

This method changes from the current directory to the nearest parent directory.

BASIC Syntax	ParentDir
Parameters	This method has no parameters.
Return Value	None

DeleteHostFile method

This method deletes a file on the host.

BASIC Syntax	DeletHostFile[*the file path*]
Parameters	*the file path*—specifies the path of the file you want to delete. (String)
Return Value	None

PrintHostFile method

This method prints a host file.

BASIC Syntax	PrintHostFile [*file path*]
Parameters	*file path*— specifies the full path of the file you want to print. (String)
Return Value	None

DelTree method

This method removes an entire local tree structure.

BASIC Syntax	DelTree[*directory path*]
Parameters	*Directory path*—specifies the path of the directory tree you want to delete. (String)
Return Value	None

PutFile method

This method puts a single file from the PC to the host. If the destination path is empty, the script sends the file to the current host working directory.

BASIC Syntax	PutFile [*source path*, *destination path*]
Parameters	*source Path*—specifies the full source path of the PC file name you want to send to the remote host. (String) *destination Path*—specifies the full destination path of the host file name. (String)
Return Value	None

DisconnectFromHost method

This method disconnects from the host.

BASIC Syntax	N/A
Parameters	This method has no parameters.
Return Value	None

PutTree method

This method sends an entire directory tree from the PC to the host. If the destination path is empty, the script uses the local default directory.

BASIC Syntax	PutTree [*source*, *destination*]
Parameters	Source—specifies the full source path of the PC file name you want to send to the remote host. (String) Destination—specifies the full destination path of the host file name. (String)
Return Value	None

FTPDIR method

This method gets the directory listing of the specified directory, with or without options in the argument. The returned text includes a full description for each file on the host. The script can process the text further if required.

BASIC Syntax	*variable* = FTPDIR [*options*]
Parameters	*options*—options are defined by the FTP server and are different from server to server.
Return Value	String

QuoteCommand method

This method enables the user to send arbitrary protocol commands to the server.

BASIC Syntax	*variable* = QuoteCommand [*command text*]
Parameters	*command text*— the protocol commands you want to send to the server.
Return Value	String

GetFile method

This method retrieves a file from the host to the PC. If the destination path is empty, the script sends the file to the current working directory.

BASIC Syntax	GetFile [*source path*, *destination path*]
Parameters	*source path*—specifies the full path to the file on the host. *Destination path*—specifies the full path to where you want to put the file on the PC.
Return Value	None

RemoteHelp method

This method returns help text from the server.

BASIC Syntax	*variable* = RemoteHelp
Parameters	(String) Remote help argument
Return Value	String

GetTree method

This method retrieves an entire directory tree structure from the host to the PC. If the destination is empty, the script uses the local default directory.

BASIC Syntax	GetTree [*source path*, *destination path*]
Parameters	*source path*—specifies the full path of the directory tree on the host. (String) *destination path*—specifies the full path to where you want to put the directory tree on the PC. (String)
Return Value	None

RemoveHostDir method

This method removes a directory from the host.

BASIC Syntax	RemoveHostDir [*directory path*]
Parameters	(*directory path*—specifies the path of the directory you want to delete. (String)
Return Value	None

HostToHostFileAppend method

This method appends a session file to another session file.

BASIC Syntax	HostToHostFileAppend [*source*, *destination*, *path*, *destination session object*]
Parameters	*source*—specifies the full source path of the files you want to append to another session file. (String) *destination*—specifies the full destination path of the file that you want to append. (String) *destination session object*—specifies the object you want to transfer. (Session object)
Return Value	None

RenameHostFile method

This method renames a file on the host.

BASIC Syntax	RenameHostFile [*sourcepath, destination path]*
Parameters	*source*—(String) path to the file to be renamed is stored *destination path*—path to where the renamed file will be stored
Return Value	None

HostToHostFileTransfer method

This method transfers a file from one session to another.

BASIC Syntax

HostToHostFileTransfer [*source path, Destination path, destination session object*]

Parameters

source Path—specifies the full source path of the file name you want to transfer to another session. (String)

destination Path—specifies the full destination path of the file name that you want to transfer. (String)

destination session object—specifies the object you want the file to be transferred to. (Session object)

Return Value

None

ReplyText method

This method returns the server reply text for the last FTP command.

BASIC Syntax	*variable* = ReplyText
Parameters	This method has no parameters.
Return Value	String

HostToHostMultipleFileTransfer method

This method transfers multiple files from one session to the other.

BASIC Syntax	*HostToHostMultipleFileTransfer [pattern, destination session object]*
Parameters	*Pattern*—specifies the pattern used by the current host to get all matching files. The pattern may be dependant on the server. *Destination session object*—specifies the object you want to transfer. (Session object)
Return Value	None

SendAccountCommand method

This method sends the account information to the server.

BASIC Syntax	SendAccountCommand [*account info*]
Parameters	*account info*—the account information you want to send to the host. (String)
Return Value	None

HostToHostTreeTransfer method

This method transfers an entire directory tree to another session.

BASIC Syntax	HostToHostTreeTransfer [source path, *Destination path, destination session object*]
Parameters	*Source Path*—specifies the full source path of the tree you want to transfer to another session. (String) *Destination Path*—specifies the full destination path of the tree that you want to transfer. (String) *Destination session object*—specifies the object you want the file to be transferred to. (Session object)
Return Value	None

ServerBanner method

This method returns the server banner. The ServerBanner method returns an error when there is no connection.

BASIC Syntax	*Variable* = ServerBanner
Parameters	This method has no parameters.
Return Value	String

LS method

This method returns the host directory listing with the specified options. The script can process the returned text if required.

BASIC Syntax	*variable* = LS [*options*]
Parameters	*options*—options are defined by the FTP server and are different from server to server.
Return Value	String

SystemType method

This method returns the name of the host’s operating system.

BASIC Syntax	*variable* = SystemType
Parameters	This method has no parameters.
Return Value	String

MakeHostDir method

This method creates a host directory.

BASIC Syntax	MakeHostDir[*directory path*]
Parameters	*Directory path*— specifies the full path of the directory you want to create. (String)
Return Value	None

UserLogin method

This method logs in to the remote server. If your server requires a password use the values in UserName, Password, Account, and InitialHostDirectory to authenticate login.

BASIC Syntax	N/A
Parameters	This method has no parameters.
Return Value	None

MGet method

This method retrieves multiple files from the host to the PC default local directory.

BASIC Syntax	MGet [*pattern*]
Parameters	pattern—specifies the pattern used by the host to get all matching file. The pattern may include wildcards native to the host. (String)
Return Value	None

WorkingHostDirectory method

This method returns the path of the current working host directory.

BASIC Syntax	*Variable* = WorkingHostDirectory
Parameters	This method has no parameters.
Return Value	String

IHclFtpSession Object properties

Account property
Connected property
DefaultSystemType property
DropExtension property
DropVersion property
FirewallPassword property
FirewallServerName property
FirewallServerPort property
FirewallType property
FirewallUserName property
ForceDefaultSystemType property
InitialHostDirectory property
Name property (IhclFtpSession)
Password property
PASVMode property
ReplyTimeout property
ServerName property
ServerPort property
SSHMode property
StripCtrlZ property
TraceFileName property
TraceMode property
TransferType property
UseFirewall property
UserName property
WriteCtrlZ property

Related Topics

IHclFtpSession Object methods

Account property

This property sends an FTP account command to the host. In order to transfer files, you must supply the account command to some hosts. The default string value is empty.

BASIC Syntax	*variable* = Account
	Account = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

Password property

This property determines the login authentication password . The default string value is empty.

BASIC Syntax	*variable* = Password
	Password = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

Connected property

This property determines if there is a connection. A TRUE value means there is a connection; a FALSE value means there is no connection. The default is FALSE.

BASIC Syntax	*variable* = Connected
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read only

PASVMode property

This property helps transfer data through any possible firewalls. The default is TRUE.

BASIC Syntax	*variable* = PASVMode
	PASVMode = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

DefaultSystemType property

This property determines the default system type. The default is UNIX. If FTP cannot detect the system type, it will use the value you set as the system default.

BASIC Syntax	*variable* = DefaultSystemType
	DefaultSystemType = *variable*
Parameters	This property has no parameters.
Data Type	TDSystem
Property Type	Read/Write

ReplyTimeout property

This property identifies the amount of time in seconds the FTP engine waits to get a reply to the command sent to the FTP server, before terminating. The default value is 120 seconds

BASIC Syntax	*variable* = ReplyTimeout
	ReplyTimeout = *variable*
Parameters	This property has no parameters.
Data Type	Long
Property Type	Read/Write

DropExtension property

This property determines if it is going to drop file extensions from file names transferred from PC to server. The default is FALSE. Use this property with mainframes (IBM, TANDEM etc) only.

BASIC Syntax	*variable* = DropExtension
	DropExtension = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

ServerName property

This property returns the name of the FTP server to which the session is connected. The default string value is empty.

BASIC Syntax	*variable* = ServerName
	ServerName = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

DropVersion property

This property determines if FTPAPI is going to drop file version numbers from VMS file names transferred to a PC. The default is FALSE.

BASIC Syntax	*Variable* = DropVersion
	DropVersion = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

ServerPort property

This property identifies the port used for FTP connections. The default port is 21.

BASIC Syntax	*variable* = ServerPort
	ServerPort = *variable*
Parameters	This property has no parameters.
Data Type	Long
Property Type	Read/Write

FirewallPassword property

This property determines the string for firewall login authentication. The default string is empty.

BASIC Syntax	*variable* = FirewallPassword
	FirewallPassword = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

SSHMode property

Determines if the FTP communications and file transfers are encrypted.

BASIC Syntax	SSHMode=*variable* *Variable*=SSHMode
Parameters	True—The FTP client encrypts data connections. The default is True False—The server does not encrypt the data connection.
Data Type	Boolean
Property Type	Read/Write

FirewallServerName property

This property determines the FTP firewall server to which the session is connected. The default string is empty.

BASIC Syntax	*variable* = FirewallServerName
	FirewallServerName = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

StripCtrlZ property

This property determines if CTRL-Z characters are stripped from the end of ASCII files transferred from PC to server. If StripCtrlZ is set to TRUE, then FTP ignores the WriteCtrlZ value. The default is FALSE.

BASIC Syntax	*variable* = StripCtrlZ
	StripCtrlZ = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

FirewallServerPort property

This property determines the server ports used to connect to the firewall. The default port is 21.

BASIC Syntax	*variable* = FirewallServerPort
	FirewallServerPort = *variable*
Parameters	This property has no parameters.
Data Type	Long
Property Type	Read/Write

TraceFileName property

This property determines the trace file name used for tracing. The default string is empty.

BASIC Syntax	*variable* = TraceFileName
	TraceFileName = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

FirewallType property

This property determines the firewall type.

BASIC Syntax	*variable* = FirewallType
	FirewallType = *variable*
Parameters	This property has no parameters.
Data Type	TFirewall
Property Type	Read/Write

TraceMode property

This property determines if the script is going to perform tracing of FTP protocol commands. The default is FALSE.

BASIC Syntax	*Variable* = TraceMode
	TraceMode = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

FirewallUserName property

This property determines the user name used to authenticate the firewall login. The default string is empty.

BASIC Syntax	variable = FirewallUserName
	FirewallUserName = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

TransferType property

This property identifies the transfer type. The default is AUTO_TRANSFER.

BASIC Syntax	*variable* = TransferType
	TransferType = *variable*
Parameters	This property has no parameters.
Data Type	TTransfer
Property Type	Read/Write

ForceDefaultSystemType property

This property forces the client to accept the default system type you define. The default is FALSE.

BASIC Syntax	*Variable* = ForceDefaultSystemType
	ForceDefaultSystemType = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

UseFirewall property

This property determines if firewall support is needed, or if the script is going to use firewall support. The default is FALSE.

BASIC Syntax	*variable* = UseFirewall
	UseFirewall = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

InitialHostDirectory property

This property identifies the directory to which the initial host directory is changed after the connection is made to the server at login. As soon as you log in, the host directory is changed automatically.

BASIC Syntax	*variable* = InitialHostDirectory
	InitialHostDirectory = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

UserName property

This property determines the login user name . The default string value is empty.

BASIC Syntax	*variable* = UserName
	UserName = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

Name property (IHclFtpSession)

This property sets and returns the session name. The default value is Session [index in the collection].

BASIC Syntax	*variable* = Name
	Name = *variable*
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

WriteCtrlZ property

This property controls whether or not a CTRL-Z end-of-file marker is inserted at the end of ASCII files being transferred from the PC to the server. A TRUE value indicates that the character is inserted. A FALSE value indicates the character is not inserted. The default is FALSE.

BASIC Syntax	*variable* = WriteCtrlZ
	WriteCtrlZ = *variable*
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

IHclFtpSessions Object

IHclFtpSessions Object creates and manages IHclFtpSession objects (a collection of Ftp sessions). IHclFtpSessions includes methods for deleting the local tree, and creating and removing sessions.

The methods and properties of this object are for use with this object and for version 6.0 or later only.

Related Topics

IHclFtpSessions Object properties

IHclFtpSessions Object methods

IHclFtpSessions Object properties

Count property

EMailAddress property

LocalDefaultDirectory property

LocalFileNamesCase property

SessionByName property

SessionByNumber property

ValidLocalFileNames property

Related Topics

IHclFtpSessions Object methods

Count property

This property returns the number of current sessions.

BASIC Syntax	*variable* = Count
Parameters	This property has no parameters.
Data Type	Long Integer
Property Type	Read Only

EMailAddress property

This property sets/returns E-mail address. The EmailAddress property is used as the password for anonymous sessions. The default string is empty.

BASIC Syntax	EmailAddress = *variable* *variable* = EmailAddress
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

LocalDefaultDirectory property

This property identifies and sets the local default directory for all sessions. If the file name or directory name does not include the full path, the script automatically sends files to or retrieves files from the default directory.

BASIC Syntax	LocalDefaultDirectory = *variable* *variable* = LocalDefaultDirectory
Parameters	This property has no parameters.
Data Type	String
Property Type	Read/Write

LocalFileNamesCase property

This property defines the case of the PC file names sent to the host. The default is lowercase.

BASIC Syntax	LocalFileNamesCase = *variable* *variable* = LocalFileNamesCase
Parameters	This property has no parameters.
Data Type	TLocalCase
Property Type	Read/Write

SessionByName property

This property returns the session object identified by the name you provide.

BASIC Syntax	*variable* = SessionByName (ByVal name as String)
Parameters	This property has no parameters.
Data Type	IhclFtpSession
Property Type	Read Only

SessionByNumber property

This property returns the session object identified by the index you provide.

BASIC Syntax	*variable* = SessionByNumber(ByVal index As Long)
Parameters	This property has no parameters.
Data Type	IhclFtpSession
Property Type	Read Only

ValidLocalFileNames property

This property creates valid local file names when files are transferrred from the host to the PC. If the value is TRUE, the script creates the valid local file names. If the value is FALSE, the script does not check for valid names.

BASIC Syntax	ValidLocalFileNames = *variable* *variable* = ValidLocalFileNames
Parameters	This property has no parameters.
Data Type	Boolean
Property Type	Read/Write

IHclFtpSessions Object methods

NewSession method

RemoveAllSessions method

RemoveSession method

RemoveSessionByName method

RemoveSessionByNumber method

Related Topics

IHclFtpSessions Object properties

NewSession method

This method creates a new FTP session with default properties.

BASIC Syntax	*variable* = NewSession
Parameters	This method has no parameters.
Return Value	IhclFtpSession object

RemoveAllSessions method

This method disconnects and removes all sessions.

BASIC Syntax	RemoveAllSessions
Parameters	This method has no parameters.
Return Value	None

RemoveSession method

This method disconnects and removes a session identified by IHclFtpSession Object.

BASIC Syntax	RemoveSession *variable*
Parameters	IhclFtpSession object
Return Value	None

RemoveSessionByName method

This method disconnects and removes a session identified by the name.

BASIC Syntax	RemoveSessionByName *variable*
Parameters	String
Return Value	None

RemoveSessionByNumber method

This method disconnects and removes a session identified by the index number.

BASIC Syntax	RemoveSessionByNumber *index*
Parameters	Long Integer
Return Value	None

Introducing WyseTerm API

WyseTerm is a communications and terminal emulation program that supports ANSI-BBS, SCO ANSI, WYSE-50, WYSE-60, VT320, and VT220 (also supports VT220 7-bit or 8-bit, VT100, or VT52) terminal modes. WyseTerm allows a user at one site to access a remote host as if the user's display station was locally attached.

You can use The WyseTerm API to customize your WyseTerm terminal emulation settings by using DDE and OLE to create dynamic links between Telnet and any other application.

DDE and OLE are mechanisms that allow applications to work together. DDE lets the applications talk to each other: one as the server and the other as the client. In Telnet, DDE can only act as a server. OLE lets the secondary application communicate with Telnet and define a set of properties and methods in it. As a result, OLE gives programmatic access to Telnet components from high-level programming languages and application scripting systems like Hummingbird Basic.

Using DDE and Telnet

Since Telnet is the server, the client application must establish a communications channel with Telnet. Communication takes place on a topic basis. This means that Telnet sets itself up as the server for a defined topic. The client then communicates with the server on a particular item of that topic. Normally, the client requests data on a particular item from Telnet, or simply makes a request that does not require the return of data.

To use DDE as a server, you must first click the Telnet DDE Server command on the Settings menu of the WyseTerm application You cannot use Telnet as a DDE client to another application. For more information on using Telnet, see Related Topics below:

Related Topics

Conversing with Telnet using DDE

Automating Telnet using OLE

Conversing with Telnet using DDE

In a DDE client/server conversation, Telnet can be used only as a DDE server. To use DDE as a server, you must first choose the Telnet DDE Server command on the Settings menu of the WyseTerm application. You cannot use Telnet as a DDE client to another application

In order for a DDE client to establish a connection with Telnet, you must specify the following in the order listed:

· The server's name (in the case of Telnet, htelnet).

· The topic it wants to have a conversation about.

You can represent this application-topic pair using the following command:

Appname|*topic*

For example,

Telnet|System

Once the client has established a connection, Telnet responds to the request transaction items issued. Some items return information, while other items perform functions. The most commonly used transaction items in Telnet include the following:

· request transaction items

· execute transaction items

Note: Check all execute transactions using the ExecReturn request transaction item.

· poke transaction items

Related Topics

Request and Execute Syntax

Request and Execute syntax

After you have started Telnet in automation mode, you can begin to configure it using request and execute items. The syntax for these items is described below.

Request Syntax

Use the following command to set a property to a value:

anyrequest=value

where *anyrequest* is any request item, and *value* is a setting you can make for that item.

Use the following command to get a request and store the value:

value=anyexecute(*parameter list*)

where *anyexecute* is any execute item and *parameter list* is the appropriate values that the execute item requires. Commas separate multiple parameters.

Execute Syntax

Use the following command to call the Telnet topic request with *value* as its parameter:

anyexecute=value

where *anyexecute* is any execute item and *value* is a setting you can make for that item.

DDE Topics Available in Telnet

If you are trying to establish a connection with the Telnet application, you must pre-specify all conversation topics at the beginning of a connection attempt. You can choose any combination of the following DDE topics in Telnet:

System—lets the client determine what topics are available and assess the format types of the messages. The System topic is a standard DDE server topic.

Terminal—lets the client control the terminal emulation mode (ANSI-BBS, SCOANSI, VT220, VT320, WYSE50, WYSE60, or DG210) and other connection settings in the Telnet program. It also lets the client load previously saved settings files.

Emulation—lets the client make changes to the settings in the Telnet program. You can set all options available on the Property tab of the Modify Terminal Setup dialog box in WyseTerm. To view these settings, open WyseTerm and click Terminal on the Settings menu.

Page Topic—lets the client control the current displayable properties such as color and attribute settings that can be changed manually in Telnet in the Modify Terminal Setup dialog box. It also allows the client to retrieve blocks of characters from anywhere on the display screen.

System Topic Request Items (HTelnet|System)

The System topic is a standard DDE server topic, which lets clients determine what topics are available and assess the format types of the messages. The System topic has the following request transaction items:

Topics item

SysItems item

Formats item

Topics Item

Lists the available topics as follows:

· System

· Terminal

· EmulationType

· Page

Syntax	Topics
Parameters	This method has no parameters.

Related Topics

System Topic Request items (HTelnet|System)

SysItems Item

Lists the available request items in the System Topic. You can choose one of the following:

· Topics item

· SysItems item

· Formats item

Syntax	SysItems
Parameters	This method has no parameters.

Related Topics

System Topic request items (HTelnet|System)

Formats Item

Lists the supported return formats. Telnet supports only text format.

Syntax	Formats
Parameters	This method has no parameters.

Related Topics

System Topic Request items (HTelnet|System)

Terminal Topic items (HTelnet|Terminal)

Emulation Type Topic items(HTelnet|EmulationType)

Terminal Topic Items (HTelnet|Terminal)

The Terminal topic lets the client control the terminal emulation mode (ANSI-BBS, SCOANSI, VT220, VT320, WYSE50, WYSE60, or DG210) and other connection settings in the Telnet program. It also lets the client load previously saved settings files. The Terminal topic includes both request and execute items.

Request Items

Terminal Topic has following request transaction items:

ConnectionStatus item

EmulationType item

ExecReturn item

Formats item

SaveData item

TopicItemList item

Execute Items

To perform an execute transaction, you must fill the data with text corresponding to the command. To confirm that the command succeeded, you must perform a request transaction using the ExecReturn item (see the preceding list) to confirm that the execute transaction was successful.

The Exec command should consist only of the command name and any parameters. Do not include any spaces.

Terminal Topic has the following Execute transaction items:

LoadSetupFile item

ConnectRlogin item

ConnectTelnet item

Disconnect item

SendString item

SendStringToTerminal item

LookForString item

StartDataSave item

StopDataSave item

ConnectionStatus Item

Returns the current state of the connection to the Telnet application. Telnet returns one of the following three states:

· connected

· connecting

· disconnecting

Syntax	ConnectionStatus
Parameters	This method has no parameters.

Related Topics

Terminal Topic items (HTelnet|Terminal)

EmulationType Item

Returns the current emulation type. Telnet supports the following types:

· ANSI-BBS

· SCIANSI

· VT220

· VT320

· Wyse50

· Wyse60

· DG210

Syntax	EmulationType
Parameters	This method has no parameters.

Related Topics

Terminal Topic items (HTelnet|Terminal)

ExecReturn Item

Returns the result of the last called Execute transaction.

Syntax	ExecReturn
Parameters	This method has no parameters.

Related Topics

Terminal Topic items (HTelnet|Terminal)

Emulation Type Topic items(HTelnet|EmulationType)

Emulation Type Topic Items (HTelnet|EmulationType)

The Emulation Type topic lets the client make changes to the settings in the Telnet program. You can set all options available on the property tab of the Modify Terminal Setup dialog box in WyseTerm. The Emulation Type topic includes both request and execute items.

Request Items

The following request transaction items exist for the Emulation Type topic.

TopicsItemList Item

Formats item

CursorPosition item

ExecReturn item

Execute Items

To perform an execute transaction, you must fill the data with text corresponding to the command. To confirm that the command succeeded, you must perform a request transaction using the ExecReturn item (see the preceding list) to confirm that the execute transaction was successful.

The Exec command should consist only of the command name and any parameters. Do not include any spaces.

The Emulation Type topic has the following Execute transaction items.

GetOption item

SetOption item

ReadKeyboardFile item

SetAttributeColor item

SetAttributeUsage item

TopicItemList Item

Lists the available request items in the Terminal Topic. You can choose one of the following:

· TopicItemList

· Formats

· EmulationType

· ConnectionStatus

· ExecReturn

· SaveData

Syntax	TopicItemList
Parameters	This method has no parameters.

Related Topics

Terminal Topic items (HTelnet|Terminal)

Emulation Type Topic items(HTelnet|EmulationType)

CursorPosition Item

Returns the current cursor position as a row, column or string. You can change the cursor position with a poke transaction to this item name. Specify the row and column position as the poke data.

Syntax	CursorPosition
Parameters	This method has no parameters.

Related Topics

Emulation Type Topic items(HTelnet|EmulationType)

GetOption Item

Gets the value of the option specified by the emulation option parameter. Generic options are available in all emulation modes. Each mode has specific options that are available only when you select that mode.

Syntax	GetOption (*emulationoption*)
Parameters	*emulationoption*—is any terminal emulation-specific setting.

Related Topics

Emulation Type Topic items(HTelnet|EmulationType)

Emulation options

Emulation Options

The GetOption() and SetOption() items require that you specify generic emulation options. Some of the options you can use with these items are listed below by category:

· Generic options—These options are available for all emulation modes.

· VT 220 and VT320 options

· Wyse50 and Wyse60 options

See the appropriate section for a complete listing of available options.

Generic Options

Option Name	Return Values
Interpret	**TRUE**–control codes are interpreted and not displayed
	FALSE–control codes are displayed and not interpreted
AutoWrap	**TRUE**–lines are automatically wrapped
	FALSE–no autowrap is used
BackSpaceDelete	**TRUE**–backspace sends delete
	FALSE--backspace sends backspace
WarningBell	**TRUE**–rings bell
	FALSE–does not ring bell
NewLine	**TRUE**–CRLF (Carriage Return - Line Feed)
	FALSE–CR (Carriage Return only)
TermStringId	**The terminal string ID**–the host uses this string to identify what type of terminal emulation the client (Telnet) is using
LocalEcho	**TRUE**–characters typed are displayed locally at the client (Telnet)
	FALSE–characters typed are sent to the host and the host determines if the client displays the character locally
Online	**TRUE**–characters are sent to the host and the terminal receives these characters
	FALSE–characters are sent to the host only
DisplayLines	**1-100**–the number of lines that can be displayed at one time
DisplayWidth80	**TRUE**–80 columns are displayed
	FALSE–132 columns are displayed
SmoothScroll	**TRUE**–scrolls smoothly
	FALSE–scrolls jump
JumpScrollNumber	**2-10**–the number of lines jumpscroll uses. Only applicable when Smooth Scroll option is FALSE.
History	**TRUE**–saves the scrolled data in a scroll back buffer
	FALSE–does not save the scrolled data
HistSize	**1-1000**–the number of lines to keep in the history buffer. Only applicable when History option is TRUE.
CaptureMode	**OFF/ON/ONHOSTCONTROL**–determines if the incoming data is captured to a file
CaptureDestination	**FILE PATHNAME**–the destination of the capture output. Only applicable when CaptureMode is ON

VT220 and VT320 Options

Option Name	Return Values
UfLock	**TRUE**–user features are locked
	FALSE–user features are not locked
UdkLock	**TRUE**–user-defined keys are locked
	FALSE–user-defined keys are not locked
EmulationMode	**VT52/VT100/VT2007bit/VT2008bit**–emulation mode is set to one of the corresponding settings
NationalSet	**North American/UK/Dutch/Finnish/French/French Canadian/German/Italian/Norwegian or Danish/Portuguese/Spanish/Swedish/Swiss**National Character Set is set to one of the corresponding settings
Multinational	**TRUE**–DEC multinational character set is used
	FALSE–only National character set is used
*UserPrefCharSetISO	**TRUE**–ISO LATIN1 is used
	FALSE–DEC multinational is used
*StatusLineMode	**OFF**–terminal information is not designated to be displayed
	ON–terminal information is designated to be displayed
	HOSTWRITEABLE–the host has been designated to change what is displayed in the status line directly

* VT320 options only.

WYSE50 and WYSE60 Options

Option Name	Return Values
ApplicationMode	**TRUE**–keys are in application mode
	FALSE–keys are in normal mode
EditModeLocal	**TRUE**–the client (Telnet) is working in local edit mode (that is, they can make local edits to a remote file)
	FALSE–the client (Telnet) is not working in local mode
BlockEndUSCR	**TRUE**–the USCR is used as the block terminator
	FALSE–the USCR character is not used as the block terminator
AutoPageMode	**TRUE**–moves to the next page automatically
	FALSE–does not move to the next page automatically
AutoScroll	**TRUE**–when text reaches the bottom of the page, the page scrolls so that the new text appears on a new bottom line
	FALSE–when text reaches the bottom of the page, the new text wraps to the top of the page
CommMode	**FULL DUPLEX/HALF DUPLEX/BLOCK**–the current communication mode is set to one of the corresponding settings
ActivePage	**0-3**–the number of Wyse pages that are active. There are four total pages
ActivePageSize	**0-100**–the page size of the active page. Note that each page can have a different size

Related Topics

GetOption item

SetOption item

Emulation Type Topic items(HTelnet|EmulationType)

SetOption Item

Sets the value of the option specified by the emulation option parameter. Generic options are available in all emulation modes. Each mode has specific options that are available only when you select that mode.

Syntax	SetOption (*emulationoption*, *value*)
Parameters	*emulationoption*—is any terminal emulation-specific setting.
	value—is the value assigned to the option.

Related Topics

Emulation Type Topic items(HTelnet|EmulationType)

Emulation options

ReadKeyboardFile Item

Reads a keyboard file in order to change the mapping of the PC key combination that is needed to forward host keys to the host.

Note: The HKeyMap program must create these keyboard files. This program can be started only in Telnet. The HKeyMap program is not DDE enabled. As a result, this item can read only existing keyboard files.

Syntax	ReadKeyboardFile(*filename*)
Parameters	*emulationoption*—is any terminal emulation-specific setting.
	filename—is the name and path of the keyboard file you want to read.

Related Topics

Emulation Type Topic items(HTelnet|EmulationType)

SetAttributeColor Item

Sets the color you want shown for the attribute you specify. The screen updates to reflect this change. The color consists of the foreground color and the background color. For example, SetAttributeColor(bold,red,blue) causes all bolded characters to display with red text and blue background. You can define the characteristics of this attribute using the SetAttributeUsage command.

Syntax SetAttributeColor(*attr*,*fgcolor*,*bgcolor*)

Parameters *attr*—is the name of the attribute.

fgcolor—is the foreground color.

bgcolor—is the background color.

Related Topics

Emulation Type Topic items(HTelnet|EmulationType)

Emulation options

SetAttributeUsage item

SetAttributeUsage Item

Sets the display of the attribute you specify. You can change the foreground color or the actual screen attribute characteristic itself. For example, you can change all bold characters to display underlined. The screen updates to reflect this change.

Syntax	SetAttributeUsage(*attr*,*realAttrib*,*realcolor*)
Parameters	*attr*—is the name of the attribute.
	realAttr—is the attribute characteristic you want to assign.
	bgcolor—is the foreground color of the attribute.

Related Topics

Emulation Type Topic items(HTelnet|EmulationType)

Emulation options

SaveData Item

Returns text that was previously saved with the StartDataSave execute item.

Syntax	SaveData
Parameters	This method has no parameters.

Related Topics

Terminal Topic items (HTelnet|Terminal)

StartDataSave

LoadSetupFile Item

Loads a Telnet settings file. If you do not specify the path, the script uses the USER directory. If the file loads, the item returns a TRUE value. If the file does not load, the script returns a FALSE value.

Syntax	LoadSetupFile *(Filename.hts)*
Parameters	*Filename.hts*—is the name of the settings file you want to load.

Related Topics

Terminal Topic items (HTelnet|Terminal)

ConnectRlogin Item

Establishes a connection using the RLOGIN protocol. Use the ConnectStatus request item to determine if the connection is successful. The item returns a TRUE value if the connection is successful and a FALSE value if the connection fails.

Syntax	ConnectRlogin *(hostname.port)*
Parameters	*hostname*—is the name of the host to which you want to connect. *port*—is the port you through which you want to make the connection.

Related Topics

Terminal Topic items (HTelnet|Terminal)

ConnectTelnet Item

Establishes a connection using the TELNET protocol. Use the ConnectStatus request item to determine if the connection is successful. The item returns a TRUE value if the connection is successful and a FALSE value if the connection fails.

Syntax	ConnectTelnet *(hostname*.port*)*
Parameters	*hostname*—is the name of the host to which you want to connect. *port*—is the port you through which you want to make the connection.

Related Topics

Terminal Topic items (HTelnet|Terminal)

Disconnect Item

Disconnects you from your current host session. Use the ConnectStatus request item to determine if the disconnection is successful. The item returns a TRUE value if the disconnection was successful and a FALSE value if the disconnection fails.

Syntax Disconnect()

Related Topics

Terminal Topic items (HTelnet|Terminal)

SendString Item

Sends the character string to the host as it appears in the *str* argument. This item returns a TRUE value if successful and a FALSE value if not successful.

Syntax	SendString (*str*)
Parameters	*str*—is any character string consisting of letters, numbers, and symbols.

Related Topics

Terminal Topic items (HTelnet|Terminal)

SendStringToTerminal Item

Sends the character string to a terminal as if sent by the host. This item sends the string as it appears in the *str* argument. This item returns a TRUE value if successful and a FALSE value if not successful.

Syntax	SendStringToTerminal (*str*)
Parameters	*str*—is any character string consisting of letters, numbers, and symbols.

Related Topics

Terminal Topic items (HTelnet|Terminal)

LookForString Item

Looks for a string (as it appears in the *str* argument) in the incoming data stream from the host. This item creates a new item with the same name as *str*. To delete the new item, set its value to an empty string. This item returns a TRUE value if successful and a FALSE value if not successful.

Syntax	LookForString (*str*)
Parameters	*str*—is any character string consisting of letters, numbers, and symbols.

Related Topics

Terminal Topic items (HTelnet|Terminal)

StartDataSave Item

Initializes the SaveData item (to blank) and starts saving the host data stream to it. This item returns a TRUE value if successful and a FALSE value if not successful.

Syntax	StartDataSave
Parameters	This method has no parameters.

Related Topics

Terminal Topic items (HTelnet|Terminal)

StopDataSave Item

Stops saving the data stream to the SaveData item. This item returns a TRUE value if successful and a FALSE value if not successful.

Syntax	StopDataSave
Parameters	This method has no parameters.

Related Topics

Terminal Topic items (HTelnet|Terminal)

Page Topic Items (HTelnet|Page)

The Page topic lets the client control the current displayable properties such as color and attribute settings. These can be changed manually in Telnet in the Modify Terminal Setup dialog box. It also lets the client retrieve blocks of characters from anywhere on the display screen.

The Page topic has the following request transaction items:

TopicItemList item

Formats item

CurrentFGColor item

CurrentBGColor item

CurrentAttribute item

CharacterAt item

CharacterAttributeAt item

CharacterFGColorAt item

CharacterBGColorAt item

CurrentFGColor Item

Determines the current foreground color given to new characters. Nocolor is the default.

Syntax	CurrentFGColor
Parameters	This method has no parameters.

Related Topics

Page Topic Items (HTelnet|Page)

CurrentBGColor Item

Determines the current background color given to new characters. Nocolor is the default.

Syntax	CurrentBGColor
Parameters	This method has no parameters.

Related Topics

Page Topic Items (HTelnet|Page)

CurrentAttribute Item

Determines the current attribute applied to new characters.

Syntax	CurrentAttribute
Parameters	This method has no parameters.

Related Topics

Page Topic Items (HTelnet|Page)

CharacterAt Item

Returns all the characters between the first and second R?C?. The top left corner of the screen has a value of r,c=1,1, and the bottom right is R,C=24,80, or R,C=24,132 depending upon the current column width. For Example, CharacterAtR1C1R24C80 returns all characters between the x, y coordinates of (1,1) and (24,80).

Note: You can access the history buffer by specifying a number between 0 and -HistorySize+1.

Syntax	CharacterAtR?C?R?C?
Parameters	This method has no parameters.

Related Topics

Page Topic Items (HTelnet|Page)

Valid Colors and Attributes

Valid Colors and Attributes

Various items require that you specify either a valid color and/or attribute as parameter of its function. In other cases, a color or an attribute may be returned as part of a transaction. The supported colors and attributes, with their numeric values, are listed below.

Note: To create a multi-characteristic attribute in C/C++, attributes are bit ORed together for the final attribute. To create a multi-characteristic attribute in Basic, they are added together to generate a final attribute.

Colors

Color	DDE String Values	OLE Decimal Values
black	Black	0
blue	Blue	1
green	Green	2
cyan	Cyan	3
red	Red	4
magenta	Magenta	5
brown	Brown	6
dark white	Dkwhite	7
grey	grey	8
light blue	light blue	9
light green	light green	10
light cyan	light cyan	11
light red	light red	12
light magenta	light magenta	13
yellow	yellow	14
white	white	15
nocolor	nocolor	16

Attributes

Attribute	DDE String Values	OLE Decimal Value	OLE Hex Value
normal	normal	0	0x0
bold	bold	1	0x01
blink	blink	2	0x02
underline	underline	4	0x04
reverse	reverse	8	0x08
protected	protected	16	0x10
hidden	hidden	128	0x20

CharacterAttributeAt Item

Returns the attribute of all characters between the first and second R?C?**.** The top left corner of the screen has a value of R,C=1,1, and the bottom right is R,C=24,80, or R,C=24,132 depending upon the current column width. For example, CharacterAttributeAtR1C1R24C80 returns the attribute of all characters between the x, y coordinates of (1,1) and (24,80).

Note: You can access the history buffer by specifying a number between 0 and –HistorySize+1.

Syntax	CharacterAttributeAt*R?c?R?C?*
Parameters	*R?*—is the number of the row.
	C?—is the number of the column.

Related Topics

Page Topic Items (HTelnet|Page)

Valid Colors and Attributes

CharacterFGColorAt Item

Returns all the foreground colors of the characters between the first and second R?C?. The top left corner of the screen has a value of R,C=1,1 and the bottom right is , c=24,80, or r,c,=24,132 depending on the current column width. For example, CharacterFGColorAtR1C1R24C80 returns the foreground color of all characters between the x,y coordinates of (1,1) and (24,80).

Note: You can access the history buffer by specifying a number between 0 and –HistorySize+1.

Syntax	CharacterFGColorAt*R?C?R?C?*
Parameters	*R?*—is the number of the row.
	C?—is the number of the column.

Related Topics

Page Topic Items (HTelnet|Page)

Valid Colors and Attributes

CharacterBGColorAt Item

Returns all the background colors of the characters between the first and second R?C?. The top left corner of the screen has a values of r,c,=1,1, and the bottom right is r,c,+24,80, or r,c=24,132 depending on the current column width.

Syntax	CharacterBGColorAt*R?C?R?C?*
Parameters	*R?*—is the number of the row.
	C?—is the number of the column.

Related Topics

Page Topic Items (HTelnet|Page)

Valid Colors and Attributes

Automating Telnet using OLE

Use Hummingbird Basic to automate Telnet using OLE. Hummingbird Basic programs, like other client programs that are OLE automation controllers, can take control of certain settings in the Telnet application.

To automate Telnet using OLE:

1. Start Telnet in OLE automation mode using Hummingbird Basic. To start Telnet in OLE you must:

· declare Telnet as the object

· create a new object, or connect to an existing one

2. Use one of the three objects available in Telnet to control some of the settings in Telnet:

· Telnet Object

· Emulation Object

· Page Object

3. Specify the appropriate property and/or method corresponding to the object you want to use.

Related Topics

Starting Telnet in Hummingbird Basic using OLE

Available OLE Objects in Telnet

Property and Method Syntax

After you have started Telnet in automation mode, you can configure properties and perform method calls. The syntax for these commands is outlined below:

Property Syntax

Use the following command to set the property value:

Telnet.*property=value*

where *property* is any property item and *value* is a setting you can make for that property.

Use the following command to get a property and store the values.

value=Telnet.*method(parameter list)*

where *method* is any method item and *parameter list* is the appropriate values that the method item requires. Commas separate multiple parameters.

All properties have a VarType. A VarType is the associate variable type in which the Property fits.

Method Syntax

Use the following command to call the Telnet object's method with value as its parameter:

Telnet.*method=value*

where *method* is any method item and *value* is a setting you can make for that method.

Related Topics

Available OLE Objects in Telnet

Available OLE Objects in Telnet

You can use OLE objects to configure Telnet settings. In Telnet, there are four OLE objects available:

Telnet Object—This object holds pre-connection and configuration management information. You can specify a settings file to use through this object, as well as determine the current emulation type in use. As well, you can send characters to the terminal and search for incoming characters using this object.

Emulation Object—This object is referenced through the Telnet Object as the LPDISPATCH emulationObject property. This object holds all emulation type information and controls the emulation-specific settings found on the Emulation-specific tab in the Modify Terminal Setup dialog box. The object contents depend on the current emulation type. Depending on which emulation type is selected, ANSI-BBS, SCOANSI, VT220, VT320, WYSE50 or WYSE60, the content of this property sheet changes.

Page Object—The Page object is referenced through the Telnet Object as the LPDISPATCH pageObject property. Use this object to control all information related to the screen and its contents.

Title Object—The Title object is referenced through the Telnet Object as the LPDISPATCH title property. This property holds all information related to the title and its contents, and is both readable and writeable.

Note: The Emulation object and the Page objects are always referenced through the Telnet object

Telnet Object

The Telnet object holds pre-connection basic configuration management information. You can specify a settings file to use through this object, as well as determine the current emulation type in use. As well, you can send characters to the terminal and search for incoming characters using this object.

The Telnet object has the creatable name of hummingbird.telnet, and consists of the following properties and methods.

Properties

EmulationType property

ConnectionStatus property

EmulationObject property

Page Object

Title Object

Methods

LoadSetupFile method

ConnectRlogin method

ConnectTelnet method

Disconnect method

Visible BOOL method

Name method

Fullname method

SendString method

SendStringToTerminal method

LookForString method

GetEventStatus method

RemoveEvent method

StartDataSave method

ReadData method

StopDataSave method

EmulationType Property

Returns the current emulation type. The following list details the types Telnet supports and the values for each type:

· ANSI-BBS—1

· SCOANSI—2

· VT220—3

· VT320—4

· WYSE50—5

· WYSE60—6

Syntax	EmulationType
Parameters	This property has no parameters.
VarType	short

Related Topics

Telnet Object

ConnectionStatus Property

Returns the current connection state of the Telnet application. The following list details the three states Telnet can return and the values for each type:

· Connected—2

· Connecting—1

· Disconnecting—0

Syntax	ConnectionStatus
Parameters	This property has no parameters.
VarType	Short

Related Topics

Telnet Object

EmulationObject Property

This property is an object reference to the Emulation object. The following list details the possible object types:

Syntax	EmulationObject
Parameters	This property has no parameters.
VarType	LPDISPATCH

Related Topics

Telnet Object

Page Object

The Page object is referenced through the Telnet Object as the LPDISPATCH pageObject property. Use this object to control all information related to the screen and its contents.

The Page Object has the following properties and methods.

Properties

FGColor

BGColor

Attribute

Methods

GetCharacter

GetCharacterString

GetCharacterAttrib

GetCharacterFGColor

GetCharacterBGColor

SetCharacter

SetCharacterAttrib

SetCharacterFGColor

SetCharacterBGColor

FGColor Property

Determines the current foreground color given to all new characters. Default is Nocolor.

Syntax	FGColor
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Page Object

BGColor Property

Determines the current background color given to all new characters. Default is Nocolor.

Syntax	BGColor
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Page Object

Attribute Property

Determines the current attribute given to all new characters. Attributes can be ORed together to generate a whole attribute.

Syntax	Attribute
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Page Object

GetCharacter Method

Returns the character from the r, c coordinates. For example GetCharacter(2,75) returns the character from the second row and the 75th column.

Syntax	GetCharacter(*short r, short c*)
Parameters	*short r*—is the number of the row where the character is located.
	short c—is the number of the column where the character is located.
VarType	UNSIGNED CHAR

Related Topics

Page Object

GetCharacterString Method

Returns the characters from the r, c coordinate pair. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, StrBuf=Telnet.GetCharString (1,1,24,80) returns all characters between the r, c coordinates of (1,1) and (24,80).

Note: To access the history buffer, specify a number between 0 and -HistorySize+1.

Syntax	GetCharString*(short r, short c, short r, short c)*
Parameters	*short r short c*—is the row and column where the string selection begins.
	short r short c—is the row and column where the string selection finishes.
VarType	UNSIGNED CHAR

Related Topics

Page Object

GetCharacterAttrib Method

Returns the attribute of a character. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, GetCharacterAttrib(1,1) returns the attribute of the character between the r, c coordinates of (1,1).

Note: To access the history buffer, specify a number between 0 and -HistorySize+1.

Syntax	GetCharAttrib(*short r*, *short c*)
Parameters	*short r*—is the number of the row where the character is located.
	short c—is the number of the column where the character is located.
VarType	UNSIGNED CHAR

Related Topics

Page Object

GetCharacterFGColor Method

Returns the foreground color of a character. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, GetCharacterFGColor(1,1) returns the foreground color of the character between the r, c coordinates of (1,1).

Note: To access the history buffer, specify a number between 0 and -HistorySize+1.

Syntax	GetCharacterFGColor*(short r, short c)*
Parameters	*short r*—is the number of the row where the character is located.
	short c—is the number of the column where the character is located.
VarType	UNSIGNED CHAR

Related Topics

Page Object

GetCharacterBGColor Method

Returns the background color of a character. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, GetCharacterBGColor(1,1) returns the background color of the character between the r, c coordinates of (1,1).

Note: To access the history buffer, specify a number between 0 and -HistorySize+1.

Syntax	GetCharacterBGColor*(short r, short c)*
Parameters	*short r*—is the number of the row where the character is located.
	short c—is the number of the column where the character is located.
VarType	UNSIGNED CHAR

Related Topics

Page Object

SetCharacter Method

Changes the character at the r, c coordinates. Changes the character at the r, c coordinates. For example, SetCharacter(1,1,"q") changes the character between the r, c coordinates of (1,1) to "q".

Note: To access the history buffer, specify a number between 0 and -HistorySize+1.

Syntax	SetCharacter*(short r, short c unsigned char ch)*
Parameters	*short r*—is the number of the row where the character is located.
	short c—is the number of the column where the character is located.
	unsigned char ch—is the character you want to set.
VarType	VOID

Related Topics

Page Object

SetCharacterAttrib Method

Sets a character attribute at the r, c coordinates. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, SetCharacterAttrib(1,1,"0x40") changes the character attribute between the r, c coordinates of (1,1) to "0x40"--a private attribute.

Note: To access the history buffer, specify a number between 0 and -HistorySize+1.

Syntax	SetCharacterAttrib(*short r, short c unsigned char attrib*)
Parameters	*short r*—is the number of the row where the character is located. *short c*—is the number of the column where the character is located. *unsigned char attrib*—is the character attribute you want to set.
VarType	VOID

Related Topics

Page Object

SetCharacterFGColor Method

Sets the foreground color of the character at the r, c coordinates. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, SetCharacterFGColor(1,1,7) changes the character's foreground color between the r, c coordinates of (1,1) to 7--a dark white color.

Note: To access the history buffer, specify a number between 0 and -HistorySize+1.

Syntax	SetCharacterFGColor(*short r, short c unsigned char color*)
Parameters	*short r*—is the number of the row where the character is located.
	short c—is the number of the column where the character is located.
	unsigned char color—is the character color you want to set.
VarType	VOID

Related Topics

Page Object

SetCharacterBGColor Method

Sets the background color of the character at the r, c coordinates. The top left corner of the screen has a value of r,c=1,1, and the bottom right is r,c=24,80, or r,c=24,132 depending upon the current column width. For example, SetCharacterBGColor(1,1,7) changes the character's background color between the r, c coordinates of (1,1) to 7--a dark white color.

Syntax	SetCharacterBGColor(*short r, short c unsigned char color*)
Parameters	*short r*—is the number of the row where the character is located.
	short c—is the number of the column where the character is located.
	unsigned char color—is the character color you want to set.
VarType	VOID

Related Topics

Page Object

Telnet Object

Title Object

The Title object is referenced through the Telnet Object as the LPDISPATCH title property. This property holds all information related to the title and its contents, and is both readable and writeable.

After the OLE client changes the title property, it cannot be changed again until you set its value to empty (null) and reset the title. If you want a blank title, use a single space.

The default title is:

Telnet *Hostname*[IP]

where *hostname* is the name of the PC to which you want to connect, and *IP* is its IP address.

Or:

Telnet

if no connection has been established.

Changing the Title

The following script changes a title:

'Sample Hummingbird Basic snippit to do this (Same for Visual Basic)

Set Term=CreateObject ("Hummingbird.Telnet")

Term.visible=1

'Telnet is now visible

Term.title="Personalized Telnet"

'do something with new title being shown

Term.title=""

'empty title reverts to default

Term.title=Chr$(32)'space character

'appears as blank

'You can get the value of the property as well:

Dim titlestring

titlestring=Term.title

LoadSetupFile Method

Loads a Telnet settings file. If you do not specify a path, the script uses the USER directory.

Syntax	LoadSetupFile(*str*)
Parameters	*str*—is the name of the settings file you want to load.
VarType	BOOL

Related Topics

Telnet Object

ConnectRlogin Method

Establishes a connection using the RLOGIN protocol. Use the ConnectStatus property to determine if the connection is successful.

Syntax	ConnectRlogin(*str host*,*str port*)
Parameters	*str host*—is the name of the host to which you want to connect.
	str port—is the port you want to use to establish the connection.
Return Type	BOOL

Related Topics

Telnet Object

ConnectTelnet Method

Establishes a connection using the TELNET protocol. Use the ConnectStatus property to determine if the connection is successful.

Syntax	*ConnectTelnet(str host,str port)*
Parameters	*str host*—is the name of the host to which you want to connect.
	str port—is the port you want to use to establish the connection.
Return Type	BOOL

Related Topics

Telnet Object

Disconnect Method

Disconnects you from your current host session. Use the ConnectStatus property to determine if the disconnection is successful.

Syntax	Disconnect()
Parameters	This method has no parameters.
Return Type	BOOL

Related Topics

Telnet Object

Visible BOOL Method

Shows or hides the Telnet window.

Syntax	Visisble BOOL
Parameters	This method has no parameters.
Return Type	True=visible False=hide

Related Topics

Telnet Object

Name Method

Returns the Telnet application file name. This is a read only command. Use this method if you have multiple applications running to determine from where the Telnet session is running.

Syntax	Name
Parameters	This method has no parameters.
Return Type	BSTR

Related Topics

Telnet Object

Fullname Method

Returns the Telnet application path and file name. This is a read only commands. Use this method if you have multiple applications running to determine from where the Telnet session is running.

Syntax	Fullname
Parameters	This method has no parameters.
Return Type	BSTR

Related Topics

Telnet Object

SendString Method

Sends the character string to the host as it appears in the STR *str* argument.

Syntax	SendString(*STR str*)
Parameters	*STR str*—is any character string consisting of letters, numbers, and symbols.
Return Type	BOOL

Related Topics

Telnet Object

SendStringToTerminal Method

Sends the character string to a terminal as if sent by the host. This item sends the string as it appears in the *STR str* argument.

Syntax	SendStringToTerminal(*STR str*)
Parameters	*Str str*—is any character string consisting of letters, numbers, and symbols.
Return Type	BOOL

Related Topics

Telnet Object

LookForString Method

Looks for a string (as it displays in the *STR str* argument) in the incoming data stream from the host. This method creates a new item with the same name as *STR*. To find the result of the search, use the event number returned, and call GetEventStatus using this number. To delete the event, use this same number with RemoveEvent.

Syntax	LookForString(*STR str*)
Parameters	*Str str*—is any character string consisting of letters, numbers, and symbols.
Return Type	SHORT EVENT—representing a number that is initially false.

Related Topics

Telnet Object

GetEventStatus Method

Returns the status of an event. To remove the event, use the event number returned with RemoveEvent.

Syntax	GetEventStatus(*short event*)
Parameters	*short event*—is the number between 0 and 65535 returned in the LookForString method..
Return Type	BOOL

Related Topics

Telnet Object

RemoveEvent Method

Removes the event for which you are looking.

Syntax	RemoveEvent(*short event*)
Parameters	*short event*—is the number between 0 and 65535 returned in the LookForString method..
Return Type	VOID

Related Topics

Telnet Object

StartDataSave Method

Starts saving the data stream for the ReadData method to read.

Syntax	StartDataSave()
Parameters	This method has no parameters.
Return Type	VOID

Related Topics

Telnet Object

ReadData Method

Reads data that you captured using the StartDataSave method.

Syntax	ReadData
Parameters	This method has no parameters.
Return Type	STRING

Related Topics

Telnet Object

StopDataSave Method

Stops saving the data stream to ReadData.

Syntax	StopDataSave
Parameters	This method has no parameters.
Return Type	VOID

Related Topics

Telnet Object

Emulation Object

The Emulation object is referenced through the Telnet Object as the LPDISPATCH emulationObject property. This object holds all emulation type information and controls the emulation-specific settings found on the Emulation-specific tab in the Modify Terminal Setup dialog box. The object contents depend on the current emulation type. Depending on which emulation type is selected, ANSI-BBS, SCOANSI, VT220, VT320, WYSE50 or WYSE60, the content of this property sheet changes.

This object contains properties and methods common to all emulation type objects. For some emulation types there are specific additional properties and methods.

Generic Properties and Methods

VT220 and VT320 Properties

WYSE50 and WYSE60 Properties

Generic Properties and Methods

The properties and methods listed below are available for all emulation modes.

Properties

Interpret property

Autowrap property

BackSpaceDelete property

WarningBell property

NewLine property

TermStringId property

LocalEcho property

Online property

DisplayLines property

DisplayWidth80 property

SmoothScroll property

JumpScrollNumber property

History property

HistorySize property

CaptureMode property

CaptureDestination property

CurrentRow property

CurrentCol property

Methods

ReadKeyboardFile method

SetAttributeColor method

SetAttributeUsage method

Interpret Property

Interprets or displays control codes. The default is TRUE.

Syntax	Interpret
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

Autowrap Property

Determines if lines wrap or not. The default is FALSE.

Syntax	AutoWrap
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

BackSpaceDelete Property

Determines if the backspace key sends a delete command or a backspace command. TRUE sends delete, and FALSE sends backspace.

Syntax	BackSpaceDelete
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

WarningBell Property

Determines if a bell rings or not. The default is TRUE.

Syntax	WarningBell
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

NewLine Property

Determines if the Return key sends a CRLF command. True sends a CRLF command, and FALSE sends a CR only. The default is TRUE.

Syntax	NewLine
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

TermStringId Property

Determines the string ID of a Terminal. The string is different for each terminal type.

Syntax	TermStringId
Parameters	This property has no parameters.
VarType	STR

Related Topics

Emulation Object

LocalEcho Property

Determines if typed characters are displayed locally by the client terminal. The default is FALSE.

Syntax	LochalEcho
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

Online Property

Determines if typed characters are forwarded to the host, or just reflected on the client terminal. The default is TRUE.

Syntax	Online
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

DisplayLines Property

Determines the number of lines used in the terminal’s display (range or 1-100). The default number of lines is 24.

Syntax	DisplayLines
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

DisplayWidth80 Property

Determines the width, in columns, of the terminal’s display. TRUE is 80 columns, and FALSE is 132 columns. The default is TRUE.

Syntax	DisplayWidth80
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

SmoothScroll Property

Determines how the terminal scrolls through data. TRUE scrolls smoothly through the data, and FALSE jump scrolls through it. The default is FALSE.

Syntax	SmoothScroll
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

JumpScrollNumber Property

Determines the number of lines that are jumped when scrolling through data. The value can be between 2-10.

Syntax	JumpScrollNumber
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

History Property

Determines if the scroll back history buffer is used. TRUE keeps the scroll back buffer, and FALSE does not. The default is TRUE.

Syntax	History
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

HistorySize Property

Determines the size of the scroll back history buffer. The number can be between 1-1000. The default is 200.

Syntax	HistorySize
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

CaptureMode Property

Determines if the incoming data is captured to a file. The values are as follows:

· 0=off

· 1=on

· 2=on with host control

Syntax	CaptureMode
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

CaptureDestination Property

Determines the destination file of the captured output.

Syntax	CaptureDestination
Parameters	This property has no parameters.
VarType	STR

Related Topics

Emulation Object

CurrentRow Property

Determines the current row on the page.

Syntax	CurrentRow
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

CurrentCol Property

Determines the current column on the page.

Syntax	CurrentCol
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

ReadKeyboardFile Method

Reads a keyboard file to change the mapping of the PC key combinations needed to forward host keys to the host.

Note: The HKeyMap program must create these keyboard files. This program can be started only in Telnet. The HKeyMap program is not DDE enabled. As a result, this item can read only existing keyboard files.

Syntax	ReadKeyboardFile(*STR filename)*
Parameters	*STR filename*—is the name and path of the keyboard file you want to read.
VarType	BOOL

Related Topics

Emulation Object

SetAttributeColor Method

Sets the foreground and background color for an attribute.

Note: Attributes can be ORed together.

Syntax	SetAttributeColor(*unsigned char attr*, *unsigned char fgcolor*, *unsighned char bgcolor*)
Parameters	*unsigned char attr*—is the attribute for which you want to set the color. *unsigned char fgcolor*—is the foreground color you want to set. *unsigned char bgcolor*—is the background color you want to set.
Return Type	VOID

Related Topics

Emulation Object

SetAttributeUsage Method

Sets the display for the attribute you specify. You can change the foreground color or the actual screen attribute characteristic. For example, you can change all bold characters to display underlined, or use any other valid attribute. The screen updates to reflect this change.

Note: Attributes can not be ORed together.

Syntax	SetAttributeUsage(*unsigned char attr*, *realAttrib*, *color*)
Parameters	*unsigned char attr*—is the name of the attribute.
	realAttrib—is the attribute characteristic you want to set.
	color—is the foreground color you want to set for the attribute.
Return Type	VOID

Related Topics

Emulation Object

VT220 and VT320 Properties

The properties listed below are available for either the VT220 or VT320 emulation modes.

VT220

UFLock property

UDKLock property

EmulationMode property

NationalSet property

Multinational property

VT320

UserPrefCharSetISO property

StatusLine property

UFLock Property

Determines if user features are locked or not. If TRUE, the user features are locked, and if FALSE, the user features are not locked.

Syntax	UfLock
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

UDKLock Property

Determines if the user-defined keys are locked or not. If TRUE, the user-defined keys are locked, and if FALSE, the user-defined keys are not locked.

Syntax	UDKfLock
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

EmulationMode Property

Determines the emulation mode. The following list details the possible emulation modes and their values:

· VT52—0

· VT100—1

· VT200 7bit—2

· VT200 8bit—3

Syntax	EmulationMode
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

NationalSet Property

Determines which National Character Set is used. The following list details the available National Character Set modes and their values:

· North American—0

· UK—1

· Dutch—2

· Finish—3

· French—4

· French Canadian—5

· German—6

· Italian—7

· Norwegian/Danish—8

· Portuguese—9

· Spanish—10

· Swedish—11

· Swiss—12

Syntax	NationalSet
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

Multinational Property

Determines if DEC multinational characters are used. If TRUE, DEC characters are used, and if FALSE, only the National Character Set is used.

Note: If this value is TRUE and UserPrefCharSetISO is TRUE, then ISO Latin 2 is the character set used.

Syntax	Multinational
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

UserPrefCharSetISO Property

Determines which character set is used. If TRUE, ISO LATIN 1 is used, and if FALSE, DEC multinational is used.

Syntax	UserPrefCharSetISO
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

StatusLine Property

Determines how terminal information displays. The following list details the possible StatusLine settings and their values:

· Off—0

· On—1

· host writable—2

Syntax	StatusLine
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

WYSE50 and WYSE60 Properties

The properties listed below are available for both the WYSE50 or WySE60 emulation modes.

ApplicationMode property

EditModeLocal property

BlockEndUSCR property

AutoPageMode property

AutoScroll property

CommMode property

ActivePage property

ActivePageSize property

ApplicationMode Property

Determines if keys are in application mode. TRUE means the keys are in application mode, and FALSE means they are not. The default mode is FALSE.

Syntax	ApplicationMode
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

EditModeLocal Property

Determines if a remote file can be edited on the client terminal. If TRUE, the client is working in local mode and can edit the file, and if FALSE, it is not in local mode and cannot edit the file. The default mode is FALSE.

Syntax	EditModeLocal
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

BlockEndUSCR Property

Determines if the USCR is used as the block terminator. If TRUE, USCR is used, and if FALSE, it is not.

Syntax	BlockEndUSCR
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

AutoPageMode Property

Determines if the client terminal moves to the next page. If TRUE, the client terminal moves to the next page automatically; if FALSE, it does not.

Syntax	AutoPageMode
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

AutoScroll Property

Determines if the page scrolls so that new text appears on a new bottom line. If TRUE, text automatically scrolls; if FALSE, text wraps to the top of the page.

Syntax	AutoScroll
Parameters	This property has no parameters.
VarType	BOOL

Related Topics

Emulation Object

CommMode Property

Determines the current communication mode. The following list details the possible communication modes and their values:

· Full Duplex—0

· Half Duplex—1

· Block—2

Syntax	CommMode
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

ActivePage Property

Determines the number of Wyse pages that are active. You can have any number of active pages between 0 and 3.

Note: Changing this value changes the state of PageObject to correspond to this new active page. As a result, PageObject is the only current active page in WYSE50 or WYSE60 mode.

Syntax	ActivePage
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

ActivePageSize Property

Determines the page size of the active page. You can have active page size in the range 0 to100.

Note: Each page can have a different size.

Syntax	ActivePageSize
Parameters	This property has no parameters.
VarType	SHORT

Related Topics

Emulation Object

Starting Telnet in Hummingbird Basic using OLE

To start Telnet in OLE automation mode, you must create an object variable and associate that variable with the application on your system.

1. Declare Telnet as the object. For example:

Dim Telnet as object

2. Start the application, if it is not already running, using the Telnet object. For example,

set Telnet=CreateObject(“Hummingbird.Telnet”)

If the application is already running, connect to it using the following statement:

set Telnet=GetObject(,"Hummingbird.Telnet")

Note: The comma preceding "Hummingbird Telnet".

3. Configure Telnet using the properties and methods included.

Related Topics

Available OLE Objects in Telnet