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

Introducing HostExplorer APIs Introducing FTP API Introducing WyseTerm API

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:

- OLE (Object Linking and Embedding) automation
- . COM (Component Object Model)
- OHIO (Open Host Interface Objects)

OLE Automation

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

For more information, see <u>OLE Automation</u>.

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.

Note: Methods and properties for TN3270 and TN5250 terminal types do not apply to Hummingbird Connectivity SecureTerm.

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 that let you embed the HostExplorer terminal into your own applications.
- Profile object
- Parser objects
- Transport objects

A <u>functional diagram</u> illustrates how 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.

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:

- 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

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 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 directory 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

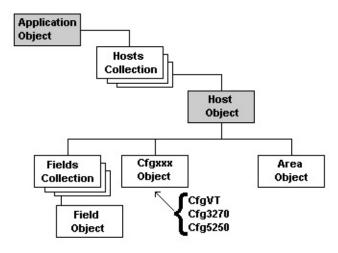
HostExplorer provides methods and properties for the following OLE objects:

- Application
- Hosts
- Host
- Area
- Field
- 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.

The following diagram illustrates the relationship between the OLE objects.



Related Topics

Methods and Properties of the Application Object

The methods and properties of the Application object let you manipulate various aspects of a HostExplorer session.

The following Application object methods and properties are available.

- CurrentHost
- <u>ExitAll</u>
 <u>GetFilePath</u>
- <u>GetCurrentDir</u>
 <u>GetProfileString</u>
 - Hosts Collection
- HostFromProfile
 - <u>HostFromShortName</u>
 <u>StartSession</u>
- Word
- WriteProfileString
- NewSession

Methods and Properties of the Hosts Object

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:

- <u>CloseAll</u>
- Count
- Item
- Next
- Open
- Note: These methods and properties are valid for all terminal types (that is, TN3270, TN5250, and TNVT).

Methods and Properties of the Host Object

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 that 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
- AllowClose
- AllowUpdates
- BFPress
- BFStatus
- Bytes
- Capture
- CaptureOIA
- Close
- Columns
- Connect
- ConnectBy
- ConnectErrorStatus
- ConnectRC
- Cursor
- CursorRC
- Device3279
- Disconnect
- <u>EAB</u>
- FieldID
- FontLarger
- FontSmaller
- GetIOBuffer
- Hide
- HideToolbar

- OIAUpdated
- Pause
- PrintScreen
- ProtectedText
- PSReserved
- PSUpdated
- PutText
- QueryCloseRequest
- QuickKeyFile
- ReceiveFile
- Restore
- Row
- Rows
- RunCmd
- RunQuickKey
- SaveScreen
- <u>SaveScrollback</u>
- Search
- SendFile
- SetFont
- ShortName
- Show
- ShowToolbar
- SilentConnect
- SystemColor

- HighlightText
- Hosts Collection
- Index
- InsertMode
- IsConnected
- IsXfer
- Keyboard
- Keys
- LoadQuickKeyFile
- LoadSlideShowFile
- <u>Maximize</u>
- Minimize
- Model
- MouseToCursor
- OIA
- SaveQuickKeyFile

- TerminalMode
- Text
- TextRC
- TN3270
- TrackMenu
- Update
- <u>WaitConnected</u>
- WaitForIO
- WaitForString
- WaitForStringRC
- WaitIdle
- WaitPSUpdated
- WaitXfer
- XferCount
- XferRC

Methods and Properties of the Area Object

The Area object provides access to a specified area of the screen. You can use the methods and properties of the Area object to write to or read from the presentation space (PS). For example, you can read to or write from the Operator Information Area (OIA) in 3270 sessions.

The following Area object methods and properties are available:

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

Methods and Properties of the Field Object

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
- Attr

ExtAttr

IsBold

IsHidden

IsModified

IsNumeric

- IsPenSelectable
- Length
- Text

- IsProtected
- Pos

Methods and Properties of the Cfg3270, Cfg5250, and CfgVT Objects

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:

- <u>Cfg3270 Objects List</u>
- Cfg5250 Objects List
- <u>CfgVT Objects List</u>

The following methods and properties are available:

- ActionOnExist
- AlwaysAutoSkip
- Answerback
- AreaCode
- AutoClearXferMonitor
- AutoCopySelectedText
- AutoMacro
- AutoUnlockKeyboard
- AutoWrap
- BellMargin
- BitMode
- BlinkToltalic
- BSIsDel
- <u>ClearAllTabStops</u>
- <u>ColorDisplay</u>
- <u>CompressBlankLinesInScrollback</u>
- ConcealAnswerback
- ConnectTimeout
- ConvertNulls
- Country
- CRToCRLF
- CursorKeyMode

- OptimizedDisplayMode
- Password
- PrintBorder
- PrintDocumentName
- PrinterDeInit
- PrinterInit
- PrintFooter
- PrintHeader
- PrintLocation
- PrintOIA
- Profile
- ProportionalFonts
- RawAddFormFeed
- RawCaptureMode
- ReplyOEM
- ReRunAutoMacro
- RespectNumeric
- RightMargin
- SaveAppend
- SaveAttrsInScrollback
- SaveConfirm
- SaveFile

- CursorMode
- CursorType
- DefaultRecvDir
- DefaultHeight
- DefaultWidth
- Dev7171Terminal
- DirectToModem
- Display3DBorder
- DisplayAttr
- DisplayControlCodes
- DisplayRowCol
- DisplayUpperCase
- EntryAssist
- FileXferProtocol
- ForceAltSize
- ForceExactSize
- Host
- <u>HostBGColor</u>
- HostColor
- HostFGColor
- KermitBinPrefix
- KermitCompression
- KermitTextMode
- KermitUseFullPath
- KeyboardProfileName
- KeypadMode
- LeftMargin
- LinesInScrollback
- Linemode
- LocalEcho
- LongName
- LUName
- Modem
- MultiLineDelete
- MultiLineInsert
- Notify
- NRCSet
- OnDisconnect
- Online

- SaveFontOnExit
- SaveMode
- SaveProfile
- SaveProfileOnClose
- ShowDialupDlg
- ShowHotspots
- ShowNulls
- ShowRecvDialog
- SmoothScrolling
- SmoothScrollSpeed
- Sound
- StatusLineMode
- TabStop
- TCPPort
- TelnetEcho
- TelnetName
- TPRINTDestination
- TypeAhead
- UPSSet
- UseDialProperties
- WindowTitle
- WordWrap
- XferBlockSize
- XferHostSytem
- XferProgramName
- XferStartAction
- XModem16BitCrc
- XModemPkt1024
- <u>XModemSendTimeout</u>
- YModemSendTimeout
- YModemUseFullPath
- ZModemAutoDownload
- <u>ZModemCrashRecovery</u>
- ZModemMaxErr
- ZModemOverwrite
- ZModemSlidingBytes
- ZModemSlidingWin
- ZModemUseFullPath

Unsupported OLE Methods and Properties

As of version 8.0, the following methods and properties no longer perform their specified actions. However, to properly maintain compatibility with earlier versions, the methods and properties continue to be members of their corresponding objects.

Application Object

DelSession method

Host Object

ShowPoppad Host property

Cfg3270, Cfg5250, and CfgVT Objects

- ALA Cfg3270 property
- ALADisplayMode Cfg3270 property
- ALAInputMode Cfg3270 property
- ALAKeyboardProfileName Cfg3270 method
- ClearScreenOnSizeChange CfgVT property
- CursorSelectMode Cfgxxxx property
- FTPApplicationName Cfgxxxx property

Related Topics

Methods and Properties of the Application Object Methods and Properties of the Host Object Methods and Properties of the Cfg3270, Cfg5250, and CfgVT Objects

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
- Conversion Group
- DDE Group
- DefType Statement
- Error Handling Group
- Flow Control Group
- Miscellaneous Group
- Operators Group
- String Group
- User Dialog Group
- Variable Info Group
- **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

- Data Type Group
- Declaration Group
- Dialog Methods and Statements
- File Group
- Math Group
- Object Group
- Settings Group
- TimeDate Group
- User Input Group

Constant Group

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

<u>General Options</u> <u>CMS-Specific Options on Upload</u> <u>TSO-Specific Options on Upload</u> <u>MUSIC-Specific Options on Upload</u>

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 File Transfer 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 <u>File Transfer Options</u> <u>CMS-Specific Options on Upload</u> <u>MUSIC-Specific Options on Upload</u>

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 File Transfer Options CMS-Specific Options on Upload TSO-Specific Options on Upload

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
@I	PF21
@m	PF22
@n	PF23
@0	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 Special Sequences Entering Control Sequences

About COM Objects

Component 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 directory where the program files are stored on your machine:

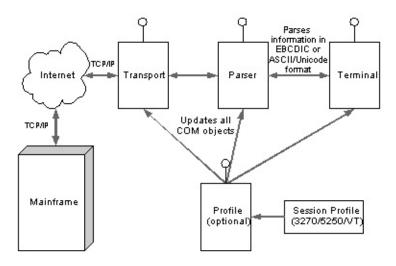
HostExplorer\SDK\Samples\COMObjects

Note: Methods and properties for TN3270 and TN5250 terminal types do not apply to Hummingbird Connectivity SecureTerm.

Related Topics About the Terminal Objects About the Profile Object About the Parser Objects About the Transport Objects

Relationship Between COM Objects

The following diagram illustrates the relationships between the Terminal, Profile, Parser, and Transport COM objects:



Related Topics About COM Objects

Unsupported COM Methods and Properties

As of version 8.0, the following methods and properties no longer perform their specified actions. However, to properly maintain compatibility with earlier versions, the methods and properties continue to be members of their corresponding objects.

Profile Object

- AllowTN3270E property (Profile interface)
- Cecp0 property (ProfileTerminal interface)
- Cecp2 property (ProfileTerminal interface)
- HostCharacterSet property (ProfileTerminal interface)
- HostKeyboard property (ProfileTerminal interface)
- VTAUPSS property (ProfileTerminal interface)
- ProportionalFonts property (ProfileFonts interface)

- IntegerName property (Profile interface)
- Cecp1 property (ProfileTerminal interface)
- Cecp3 property (ProfileTerminal interface)
- HostCodePage property (ProfileTerminal interface)
- Language property (ProfileTerminal interface)
- ShowHotspots property (ProfileDisplay interface)
- ProfileSchemes interface

Transport Object

- AddFeature method
- RemoveFeature method

Parser Object Methods

- GetCecp
- SetCecp

Parser Object Properties

- BellMargin
- Columns
- EnableAutoDeleteFromNextField
- EnableAutoNextField
- EnableDisplayRowColumnOnOIA
- GraphicsCursorType
- Language
- MaxUndoRedoEvents
- RightMargin
- ScrollStart
- VTScrollSpeed
- **Related Topics**

Renamed or Moved COM Methods, Properties, and Interfaces

- CharSet
- DetectChainedIO
- EnableAutoInsertToNextField
- EnableDisplayHostAddressOn OIA
- EnableOEMReply
- InsertKeyStyle
- LeftMargin
- Password
- Rows
- VTNRC

Renamed or Moved COM Methods, Properties, and Interfaces

For version 8.0, certain methods, properties, and interfaces have been renamed or moved.

Profile Object

The following changes have been made to the Profile object:

- several interfaces have been renamed
- several properties have been renamed
- several properties have been moved from one interface to another

Transport Object

The following properties of the Transport object have been renamed or implemented differently for 8.0:

Former Name	New Name or Implementation
DeviceName	LUNameRequested
Feature	SetFeature and GetFeature methods
KerberosUsername, KerberosAlternateUsername, and KerberosVersion	SetKerberosInfo
LockOnAttention	HOSTEX_LOCK_ON_ATTENTION value of <u>HEPARSER_FEATURE</u> Data Type

Parser Object

The following properties of the Parser objects have been renamed or implemented differently for 8.0:

Former Name	New Name or Implementation
CellDelimited	<u>CellCopyMode</u>
Feature	GetFeature and SetFeature methods
NextFieldKey	OnPasteFieldModeTabCharacter
NRC	NRCID
ReplaceFieldAttributeWith	OnCopyReplaceFieldAttributeWith
VTUPSS	UPSS
WrapLines	AutoWrap
XferMode	TransferMode
XferErrorCode	TransferErrorCode

Related Topics Unsupported COM Methods and Properties

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 Profile 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

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

About the Profile Object

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

- Profile Interface
- ProfileGraphics Interface
- ProfileMouse Interface
- ProfileTrackMenu Interface
- ProfileVTCharset Interface
- ProfileColor Interface
- ProfileSecurity Interface
- ProfileCursor Interface
- ProfilePCPrint Interface
- ProfilePrintSession Interface
- ProfileHostPrinting Interface
- ProfileEvents Interface
- ProfileSessionWindow Interface

- ProfileTerminal Interface
- ProfileKeyboard Interface
- ProfileToolbar Interface
- ProfileTranslationTable Interface
- ProfileEdit Interface
- ProfileFileTransfer Interface
- ProfileDisplay Interface
- ProfileFonts Interface
- ProfilePrintScreen Interface
- ProfileCapture Interface
- ProfileHotspots Interface
- ProfileConnection Interface
- ProfileSound Interface

Profile Interface

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

Methods

The Profile interface consists of the following methods:

- Load
- LoadColorScheme
- LoadFileTransferScheme
- LoadQuickKeyFile
- Save

The Profile interface consists of the following properties:

- AddOIAToCapture
- AllowEmuTracing
- AllowErrorRestart
- AttnFormat
- Capture
- <u>ClearPassword</u>
- <u>Color</u>
- <u>Connection</u>
- Cursor
- DDEServerName
- Display
- Edit
- EmuTraceFilename
- EnableHLLAPITracing
- EnableTracing
- Events
- FileTransfer
- Fonts
- GlobalSettingsPath
- Graphics
- HLLAPITraceFilename
- HostPrinting

- Hotspots
- Keyboard
- KillMacrosOnSessionExit
- Mouse
- PCPrint
- PrintSession
- PrintScreen
- ProfileName
- QueryShutdown
- RecordPortableMacros
- ReRunAutoMacro
- Security
- SessionWindow
- Sound
- Terminal
- TerminalType
- Toolbar
- TrackMenu
- TranslationTable
- UserDirectory
- VTCharset
- WinDDEEnabled

ProfileTerminal Interface

The ProfileTerminal interface lets you set configuration settings related to the terminal.

The ProfileTerminal interface consists of the following properties:

- AlternateScreen
- CharacterSet
- CustomModel
- CustomModelCols
- CustomModelRows
- DetectChainedIO
- ForceAltSize
- New3270EAB
- NewModel3279
- NewModelType
- ReplyOEM
- ShortName
- VT8BitMode
- VTAnswerback
- VTWrapLine

- VTAutoResize
- VTBSIsDel
- VTConcealAnswerback
- VTDefColsPerScreen
- VTDefLinesPerScreen
- VTDisplayMode
- VTEnableSSH
- VTForce8Bit
- VTLocalEcho
- VTNewTerminalType
- VTOnLine
- VTScrollSpeed
- VTSmoothScroll
- VTTerminalID

ProfileGraphics Interface

The ProfileGraphics interface lets you set configuration settings related to graphics.

The ProfileGraphics interface consists of the following properties:

- <u>APL</u>
- GraphicsCursorType
- GraphicsModel
- LightPen
- ProgramSymbols
- PSCellSize

ProfileKeyboard Interface

The ProfileKeyboard interface lets you set configuration settings related to the keyboard.

The ProfileKeyboard interface consists of the following properties:

- AllowAIDKeyRepeat
- CurrentKeyboard
- LockOnAttention
 MapNumLock
- RemapKeypad
- VTEnableBreak
- VTNewLineMode

- AllowDiac
- KeyboardType
- TypeAhead
- <u>TypeAheadTimeout</u>
 <u>VTCursorKeyApplMode</u>
 - VTKeypadApplMode

ProfileMouse Interface

The ProfileMouse interface lets you set configuration settings related to the mouse.

The ProfileMouse interface consists of the following properties:

- BlockSelect
- SelectHilight

ProfileToolbar Interface

The ProfileToolbar interface lets you set configuration settings related to the toolbar.

The ProfileToolbar interface consists of the following properties:

- BigToolbar
- Button NumTools
- MaxBitmaps
- <u>ShowTips</u>
 <u>TBUserBitmaps</u>
- <u>ToolbarDockType</u>
 <u>ToolbarFilename</u>

ProfileTrackMenu Interface

The ProfileTrackMenu interface lets you set configuration settings related to the Track Menu.

The ProfileTrackMenu interface consists of the following properties:

- TrackCommands
- TrackLabels

ProfileTranslationTable 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.

The ProfileTranslationTable interface consists of the following property: <u>CurrentLanguage</u>

ProfileVTCharset Interface

The ProfileVTCharset interface lets you set configuration settings related to the VT Character Set.

The ProfileVTCharset interface consists of the following properties:

- <u>VTForceNRC</u>
- VTNRC
- VTNRCMode
- <u>VTUPSS</u>

ProfileEdit Interface

The ProfileEdit interface lets you set configuration settings related to editing.

Methods

The ProfileEdit interface contains the following methods:

- ClearAllTabStops
- TabStop

The ProfileEdit interface consists of the following properties:

- <u>AlwaysAutoskip</u>
- <u>AutoCopyKeepSelection</u>
- CellDelimited
- <u>ClipFormatCSV</u>
- ClipFormatPasteLink
- <u>ClipFormatText</u>
- <u>CSVCopyEmptyFields</u>
- CutChar
- EntryAssist
- LeftMargin
- MultiLineDelete
- NoLockKeyb
- PasteChar
- RemoveTrailingBlankOnCopy
- RespectNumeric
- <u>SmartInsert</u>
- WordWrapWithNewLineReturn

- AutoCopy
- BellMargin
- <u>ClipFormatBitmap</u>
- ClipFormatHE
- ClipFormatRTF
- ConvertNulls
 - <u>CSVTrimFields</u>
 - CutMode
 - InsertResetByAttn
 - MoveCursorAfterPaste
 - MultiLineInsert
 - NumericCharacters
 - PasteMode
 - ResetMDTOnEraseInput
 - RightMargin
 - WordWrap

ProfileColor Interface

The ProfileColor interface lets you set configuration settings related to color.

Methods

The ProfileColor interface contains the following method:

<u>SystemColor</u>

The ProfileColor interface consists of the following property:

<u>Schemes</u>

ProfileFileTransfer Interface

The ProfileFileTransfer interface lets you set configuration settings related to transferring files.

The ProfileFileTransfer interface consists of the following properties:

- Append
- ASCII
- AutoCC
- <u>AutoClearMonitor</u>
- BlkSize
- CRLF
- CustomTransferTable
- DefaultDownloadPath
- DefaultProtocol
- DefaultRecvDir
- DefaultUploadPath
- DownloadHostFileName
- DownloadPCFileName
- DownloadTranslate
- ExtraOptions
- FileExistAction
- Host
- INDFILEName
- KmBinaryPrefix
- KmRLE
- KmTextMode
- KmUseFullPath
- Lrecl
- QuickMode
- Recfm
- Schemes

- <u>ShowRecvDlg</u>
- UploadHostFileName
- UploadPCFileName
- UploadTranslate
- UserDefinedDownload
- UserDefinedUpload
- VTAutoClearMonitor
- XferBlockSize
- XferDest
- XferHostCodePage
- XferPCCodePage
- XferSource
- XferStartAction
- Xm1KPacket
- <u>XmAckTimeout</u>
- XmCRC
- YmAckTimeout
- YmUseFullPath
- <u>ZmAutoDownload</u>
- <u>ZmCrashRecovery</u>
- ZmMaxErrors
- ZmOverwriteMngmt
- <u>ZmSlideWindow</u>
- ZmUseFullPath
- ZmWindowSize

ProfileSecurity Interface

The ProfileSecurity interface lets you set configuration settings related to security.

The ProfileSecurity interface consists of the following properties:

- Kerberos
- KerberosAltName
- KerberosEncryption
- KerberosForwardTkt
- KerberosVersion
- <u>SecurityOption</u>

ProfileDisplay Interface

The ProfileDisplay interface lets you set configuration settings related to screen display.

The ProfileDisplay interface consists of the following properties:

- BlinkToltalic
- DisplayInOIA
- DisplayUpperCase
- StatusLineMode
- VTHostWritableStatusLine
- <u>VTMaxScrollBufferSize</u>
 <u>VTResetISOColors</u>
- VTSaveAttribsInScrollback
 VTSaveEraseScreens
- ColumnSeparators
- DisplayRowCol
- ShowNulls
 - VTClearScreenOnSizeChange
 - VTISOColors
- VTScrollNoBlanks

ProfileCursor Interface

The ProfileCursor interface lets you set configuration settings related to the cursor.

The ProfileCursor interface consists of the following properties:

- <u>CursorMode</u>
- <u>CursorType</u>
- <u>DisplayCrossHairCursor</u>
- <u>MoveCursorOnMouseClick</u>

ProfileFonts Interface

The ProfileFonts interface lets you set configuration settings related to fonts.

Methods

The ProfileFonts interface has the following method:

<u>SetFont</u>

The ProfileFonts interface consists of the following properties:

- CharacterSpacing
- VariableWidthFont

ProfilePCPrint Interface

The ProfilePCPrint interface lets you set configuration settings related to the PCPRINT program.

The ProfilePCPrint interface consists of the following properties:

- PrintMode7171
- PrinterDeinit
- PrinterInit
- TprintMode

ProfilePrintScreen Interface

The ProfilePrintScreen interface lets you set configuration settings for printing a screen.

The ProfilePrintScreen interface consists of the following properties:

- AddFormFeed
- DisplayPrintDlg
- PrintBlackAndWhite
- PrinterDocname
- PrinterHeader
- PrintOIA
- PrintScreenFontName
- <u>PRTSCRUseSpecificPrinter</u>

- DisplayAbortDlg
- HostScreenPerPage
- PrintBorder
- PrinterFooter
- PrintLocation
- PrintReversedColors
- PrintScreenFontPointSize

ProfilePrintSession Interface

The ProfilePrintSession interface lets you set configuration settings related to general output.

The ProfilePrintSession interface consists of the following properties:

- HostName
- LimitToSingleInstance
- LUName
- LUType
- ProfileName
- <u>StartPrinter</u>
- StopPrinter

ProfileCapture Interface

The ProfileCapture interface lets you set configuration settings related to the saving of files.

The ProfileCapture interface consists of the following properties:

- SaveAppend
- SaveConfirm
- SaveFileName
- SaveMode
- VTCaptureMode

ProfileHostPrinting Interface

The ProfileHostPrinting interface lets you set configuration settings related to host printing.

The ProfileHostPrinting interface consists of the following properties:

- VTAutoFormFeed
- VTiPrintMaxRows
- VTPrintByPassWindows
- VTPrintDisableTranslation
- VTPrinterEnableTimeout
- VTPrinterInit
- VTPrinterTimeoutValue
- VTPrintFileMode
- VTPrintLFtoCRLF
- VTUseSpecificPrinter

- VTiPrintMaxCols
- VTPassThruUPSS
- VTPrintDefaultFont
- VTPrinterDeinit
- VTPrinterFontName
- VTPrinterTimeout
- VTPrintFile
- VTPrintFitFontToPage
- VTPrintTarget

ProfileHotspots Interface

The ProfileHotspots interface lets you configure settings related to hotspots on the host screen.

The ProfileHotspots interface consists of the following properties:

- DisplayStyle
- EnableHotspots
- MouseActivation
- Schemes

ProfileEvents Interface

The ProfileEvents interface lets you configure settings related to programmed events.

Methods

The ProfileEvents interface has the following method:

<u>RemoveEvent</u>

The ProfileEvents interface consists of the following properties:

- EnableEvents
- Schemes

ProfileConnection Interface

The ProfileConnection interface lets you configure settings related to the connection to the host.

The ProfileConnection interface consists of the following properties:

- <u>acPassword</u>
- <u>AlwaysPromptForHostName</u>
- AreaCode
- AutoEndMacroName
- AutoMacroName
- AutoRunMacroDelayTime
- AutoSignOn
- BackspaceKeyInterpretation
- ConnectBy
- Country
- CountryCode
- CountryID
- DeviceName
- DirectToModem
- EnableEMode
- EnableInfiniteRetries
- EnterKeyInterpretation
- HostName
- KeyboardBufferMode
- LUName

- Modem
- ModemID
 - <u>NumberOfRetries</u>
 - Password
 - Port
 - PortList
 - RetryDelayTimeBetweenHosts
 - <u>ShowDialupDlg</u>
 - SilentConnect
 - SYSREQasIACIP
 - TelnetEcho
 - TelnetName
 - Timeout
 - UponDisconnect
 - UseDialProp
 - UserName
 - VTDoHostWindowSize
 - VTInitiateTelnetNegotiation
 - VTLineMode

ProfileSessionWindow Interface

The ProfileSessionWindow interface lets you configure settings for the session window.

The ProfileSessionWindow interface consists of the following properties:

- DisplayBorder
- EnableWorkspaceBackgroundBitmap
- FullScreenMode
- KeepFontAspectRatio
- LongName
- PromptOnClose
- ResizeBehavior
- SaveProfOnClose

- SaveFontOnExit
- SnapFrameBack
- SwitchScreenType
- WindowState
- WindowTitle
- WorkSpaceBackgroundBitmap
- WorkspaceBackgroundColor
- WorkspaceForegroundColor

ProfileSound Interface

The ProfileSound interface lets you set configuration settings related to program sounds.

The ProfileSound interface consists of the following properties:

- Notify
- Sound

Data Types of the Profile Object

The Profile object contains the following data types:

- HOSTEX_ATN_FORMAT Data Type
- HOSTEX_BACKSPACE_KEY_INTERPRETATION Data Type
- HOSTEX_CELL_DELIMITED Data Type
- HOSTEX_CONNECT_BY Data Type
- HOSTEX_CUT_MODE Data Type
- HOSTEX_ENTER_KEY_INTERPRETATION Data Type
- HOSTEX_FIELD_ATTR_REPLACEMENT Data Type
- HOSTEX_GRAPHICS_CELLSIZE Data Type
- HOSTEX_GRAPHICS_CURSOR_TYPE Data Type
- HOSTEX_GRAPHICS_MODEL Data Type
- HOSTEX_HOTSPOT_DISPLAY Data Type
- HOSTEX_HOTSPOT_MOUSE_ACTIVATION Data Type
- HOSTEX_KEYBOARD_BUFFER_MODE Data Type
- HOSTEX_KEYBOARD_TYPE Data Type
- HOSTEX_LINEMODE Data Type
- HOSTEX_NEXT_FIELD_KEY Data Type
- HOSTEX_OIA_DISPLAY Data Type
- HOSTEX_PASTE_MODE Data Type
- HOSTEX_PRINT_TARGET Data Type
- HOSTEX_PRINTFILE_MODE Data Type
- HOSTEX_RESIZE_BEHAVIOR Data Type
- HOSTEX_SAVE_OPTIONS Data Type
- HOSTEX_SECURITY_OPTIONS Data Type
- HOSTEX_SELECTION_MODE Data Type
- HOSTEX_STATUS_LINE_MODE Data Type
- HOSTEX_SWITCHSCREENTYPE Data Type
- HOSTEX_TELNETECHO Data Type
- HOSTEX_TPRINT_OUTPUT Data Type

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 <u>HEPAR3270 and HEPAR5250 objects</u>
- both the <u>HEPAR3270 and HEPARVT objects</u>
- only the <u>HEPARVT object</u>

There are also methods, properties, and data types common to <u>all three</u> <u>objects</u>.

Methods of the Parser Objects

The following are methods of the Parser objects:

- AsciiToHost
- ClearSel
- FindString
- GetCursorPosition
- GetFeature
- GetFieldAttribute
- GetFieldCount
- GetFieldExtAttribute
- <u>GetFieldIndex</u>
- GetFieldLength
- GetFieldPos
- GetFieldText
- GetScreenText
- GetSel
- <u>GetSelectionArea</u>
- GetValue
- HostToAscii
- IsFieldBold
- IsFieldHidden
- IsFieldModified
- IsFieldNumeric
- IsFieldPenSelectable
- IsFieldProtected
- MoveCursorRelative

- PasteDataToScreen
- PutString
- PutText
- ReceiveFile
- ReplaceSel
- SendAid
- SetCursorPosition
- SetFeature
- SetFieldText
- SendFile
- SendKeys
- SetSel
- SetValue
- WaitConnected
- WaitForCursor
- WaitForCursorMove
- WaitForIO
- WaitForString
- WaitHostQuiet
- WaitIdle
- WaitPSUpdated
- WaitXfer
- WriteProtectedText

Properties of the Parser Objects

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

- Answerback
- APLInputMode
- AutoWrap
- BufRows
- CanChangeScreen
- CaptureMode
- CellCopyMode
- ConnectBy
- ConnectErrorStatus
- ConnectRC
- ConvertNulls
- CutMode
- EnableAPL
- EnablePrinterTimeout
- <u>GraphicsModel</u>
- HistoryLines
- HLLAPIName
- <u>HostResponseTime</u>
- <u>HostWritableString</u>
- KeyboardLocked
- ModelColumns
- ModelRows
- MoveCursorOnMouseClick
- NRCID
- NumericCharacters
- NVTMode
- OIAString
- OIAStringW
- OnCopyReplaceFieldAttributeWith

- OnPasteFieldModeTabCharacter
- PasteMode
- PrintByPassWindows
- PrinterDeInitString
- PrintDisableTranslation
- PrinterInitString
- PrinterTimeout
- PrinterTimeoutValue
- PrintLFtoCRLF
- ProgramSymbols
- <u>SaveAppend</u>
- SaveFileName
- ScreenChanged
- ScreenCol
- ScreenRow
- SelectionMode
- <u>SessionName</u>
- SoftCharacterSetID
- StatusLineMode
- TerminalID
- TerminalModel
- Text
- Transport
- TypeAheadTimeout
- ∎ <u>UPSS</u>
- ValidateNumericFieldData
- TransferErrorCode
- TransferMode

Data Types of the Parser Objects

The Parser objects contain the following data types:

- <u>HEPARSER_FEATURE Data Type</u>
- HOSTEX_CAPTURE_MODE Data Type
- HOSTEX_CELL_DELIMITED Data Type
- HOSTEX_CONNECT_BY Data Type
- HOSTEX_CUT_MODE Data Type
- HOSTEX_FIELD_ATTR_REPLACEMENT Data Type
- HOSTEX_GRAPHICS_MODEL Data Type
- HOSTEX_INSERT_KEY_STYLE Data Type
- HOSTEX_NEXT_FIELD_KEY Data Type
- HOSTEX_PASTE_MODE Data Type
- HOSTEX_SELECTION_MODE Data Type
- HOSTEX_STATUS_LINE_MODE Data Type
- HOSTEX_TERMINAL_ID Data Type
- HOSTEX_TERM_MODEL Data Type
- HOSTEX_TRANSFER Data Type

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 are:

- 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 <u>HETP5250 object</u>
- both the <u>HETP3270 and HETP5250 objects</u>
- only the <u>HETPVT object</u>
- both the <u>HETP3270 and HETPVT objects</u>

There are also methods and properties common to the <u>HETP3270</u>, <u>HETP250</u>, and <u>HETPVT objects</u>.

Related Topics Methods of the Transport Objects Properties of the Transport Objects Data Types of the Transport Objects

Methods of the Transport Objects

The following properties are methods of the Transport objects:

- GetFeature
- GetStatusString NegotiateNAWS SendData
- SendFunctionKey
- SendKeepAlive
- SetFeature
- <u>SetHostPrintTransformInfo</u>
- SetKerberosInfo ToggleBlockReceive

Related Topics Properties of the Transport Objects Data Types of the Transport Objects

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
- LUNameRequested
- DeviceType
- EnableEMode
- EnableSSH
- EnableTracing
- HostAddress
- HostName
- IsEncrypted
- IsReceiveBlocked
- Keyboard
- LineMode
- MaxBlockSize
- MessageQueueLibrary
- MessageQueueName
- ModelColumns

- ModelRows
- NumberOfRetries
- Password
- PerformingTransfer
- Port
- PortList
- RetryDelayTimeBetweenHosts
- <u>SecurityOption</u>
- SessionKeepAlive
- TelnetEcho
- TelnetIsLineMode
- TelnetIsLocalEcho
- TelnetName
- TerminalModel
- TerminalOnline
- TerminalType
- TNESession
- TraceFilename
- Username

Related Topics

Methods of the Transport Objects Data Types of the Transport Objects

Data Types of the Transport Objects

The Transport objects contain the following data types:

- <u>HETRANSPORT_FEATURE Data Type</u>
- HOSTEX_ATN_FORMAT Data Type
- HOSTEX_CON_STATUS Data Type
- HOSTEX_DEVICE_TYPE Data Type
- HOSTEX_ENCRYPTED Data Type
- HOSTEX_FUNCTION_KEY Data Type
- HOSTEX_LINEMODE Data Type
- HOSTEX_SECURITY_OPTIONS Data Type
- <u>HOSTEX_TELNETECHO Data Type</u>
- HOSTEX_TERM_MODEL Data Type
- HOSTEX_TOGGLE_RECEIVE Data Type

Related Topics

Methods of the Transport Objects Properties of 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.

The OhioManager interface consists of the following methods:

- <u>CloseSession</u>
- OpenSession

The OhioManager interface consists of the following properties:

- OhioVersion
- Sessions
- VendorName
- VendorObject
- VendorProductVersion

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.

The OhioSessions interface consists of the following method: <u>Refresh</u>

The OhioSessions interface consists of the following properties:

- Count
- Item

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)

The OhioSession interface consists of the following methods:

- Connect
- Disconnect
- isConnected
- OnSessionChanged

- <u>CanChangeScreen</u>
- ConfigurationResource
- Screen
- <u>SessionName</u>
- SessionType

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.

The OhioScreen interface consists of the following methods:

- FindString
- <u>GetData</u>
 <u>OnScreenChanged</u>
- OnCursorMoved
- <u>OnSizeChanged</u>
 <u>PutString</u>
- SendKeys
- WaitForString
- WaitForInput
- WaitIdle

The OhioScreen interface consists of the following properties:

- <u>Columns</u>
 <u>Cursor</u>
- <u>Fields</u> <u>OIA</u>
- <u>Rows</u>
 <u>String</u>

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.

The OhioOIA interface consists of the following properties:

- Alphanumeric
- <u>CommCheckCode</u>
 <u>InputInhibited</u>
- InsertMode
 - MachineCheckCode
- Numeric
- Owner

APL

ProgCheckCode

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.

The OhioFields interface consists of the following methods:

- FindByPosition
- FindByString
- Refresh

The OhioFields interface consists of the following properties:

- Count
- Litem

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.

The OhioField interface consists of the following method:

<u>GetData</u>

The OhioField interface consists of the following properties:

- Attribute
- HighIntensity Length
- <u>Modified</u>
 <u>Normal</u>

End

- Numeric
- PenSelectable Start Protected
- String

OhioPosition Interface

The OhioPosition interface provides the row and column coordinates of the cursor position. You can create an OhioPosition by using the <u>CreateOhioPosition</u> method.

The interface is used by the following sub-interfaces:

- <u>OhioScreen Interface</u>
- <u>OhioFields Interface</u>
- OhioField Interface

The OhioPosition interface consists of the following method:

CreateOhioPosition

The OhioPosition interface consists of the following properties:

- Column
- Row

Data Types of OHIO

Ohio contains the following read-only data types:

Ohio interface

OHIO_DIRECTION Data Type

OhioSession interface

- OHIO_STATE Data Type
- OHIO_TYPE Data Type

OhioScreen interface

- <u>OHIO_COLOR Data Type</u>
- <u>OHIO_EXTENDED Data Type</u>
- <u>OHIO_FIELD Data Type</u>
- <u>OHIO_PLANE Data Type</u>
- <u>OHIO_UPDATE Data Type</u>

OhioOIA interface

- <u>OHIO_INPUTINHIBITED Data Type</u>
- OHIO_OWNER Data Type

About Legacy APIs

HostExplorer provides the following existing or "legacy" APIs:

- <u>EHLLAPI (Extended High Level Language Application Programming Interface) and WinHLLAPI (Windows HLLAPI)</u>—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.

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 Module
- WinHLLAPI Module

The EHLLAPI interface includes:

- a new Window Close (201) function.
- an extended ConnectPS (1) function.

Related Topics Irma Compatibility Mode Configuration Tips

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
- <u>Yield Wait</u>
 Auto Sync
- <u>Return Extra Session Info</u>
- Allow Connect Physical
- <u>Update Screen After Copy</u>
- Convert Nulls

Related Topics EHLLAPI Support in VT and NVT Modes EHLLAPI Development Files NVT Mode Functions NVT Mode Exceptions

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

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

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
- <u>EHLLAPIConvertPosToRowCol</u>
- <u>EHLLAPIConvertRowColToPosition</u>
- EHLLAPICopyFieldToString
- EHLLAPICopyOIA
- EHLLAPICopyPS
- EHLLAPICopyPSToString
- EHLLAPICopyStringToField
- EHLLAPICopyStringToPS
- EHLLAPIDisconnect
- EHLLAPIFindFieldPosition
- <u>EHLLAPIGetRowString</u>
- EHLLAPIGetVersion
- EHLLAPIPause
- EHLLAPIQueryCursorLocation

- EHLLAPIQueryFieldAttribute
- EHLLAPIQuerySessions
- EHLLAPIQuerySessionStatus
- EHLLAPIReceiveFile
- EHLLAPIRelease
- EHLLAPIReserve
- EHLLAPIReset
- EHLLAPISearchField
- EHLLAPISearchPS
- EHLLAPISendFile
- EHLLAPISendKey
- EHLLAPISetCursorLocation
- <u>EHLLAPISetSessionParameters</u>
- EHLLAPIWait

Related TopicsEHLLAPI and WinHLLAPI DLL SupportEHLLAPI Development FilesVisual Basic Interface

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

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 is 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

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 EHLLAPI and WinHLLAPI Flags EHLLAPI Support in VT and NVT Modes WinHLLAPI 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 TopicsSpecial EHLLAPI and WinHLLAPI FlagsEHLLAPI Development FilesEHLLAPI Support in VT and NVT Modes

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.

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.

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

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, and 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?

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 Message
- Execute Message
- Poke Message
- Request Message

To have a DDE conversation, you need the following fields:

- Application Name Field
- Topic Field

Related Topics
System Topic

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:

- Microsoft Excel »

Related Topics How Does DDE Work? DDE Sample Code System Topic

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", "Informat. 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) * iNumCol))
        "L" + LTrim$(Str$(iNumCols))
        Data$ = Data$ + DDERequest$(iChanNum, request$
      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

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?

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 <u>Introducing FTP OLE API</u>. 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.

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 create your own script files.

To open the source file (.ebs) in Hummingbird Basic Workbench, click Open on the File menu. You can print the loaded source file by clicking the Print command on the File menu. If you want to run the loaded source file, click 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.

Introducing FTP OLE API

The FTP API is an 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.

OLE Automation

OLE Automation is a Windows facility that permits data exchange between applications and automates tasks. When an object, such as an image file created with a paint program, is linked to a compound document, such as a spreadsheet or a document created with a word processing program, the document contains only a reference to the object. Any changes made to the contents of a linked object are seen in the compound document.

You can use OLE Automation to access and control FTP for Windows Explorer. You can write OLE Automation clients using a variety of tools, such as Hummingbird Basic, Visual Basic, C++, and Java.

The name of the Automation object is HclFtp.Engine.

Related Topics Creating an OLE Script Sample Client Source Code

Creating an OLE Script

You can code OLE Automation containers to implement all the features and functions of FTP in another application that uses OLE features, such as Hummingbird Basic. You can work with FTP session objects to call functions, such as connecting to a host, transferring files from host to host, and so on. You can use any tool that supports OLE Automation control, such as Visual C++ and Visual Basic.

To create an OLE script:

1. Create the main FTP Engine Object. All objects support a dual interface. This lets you fully use the FTP OLE features. For example:

Dim FtpEngine As Object
Set FtpEngine = CreateObject("HclFtp.Engine")

2. Retrieve an FTP Sessions collection. This lets you set such things as local drives, access permissions, and so on. For example:

Dim FtpSessions As Object
Set FtpSessions = FtpEngine.Sessions

3. Create the FTP Session object. For example:

Dim FtpSession1 As Object
Set FtpSession1 = FtpSessions.NewSession

4. Set properties, such as server name, user name, user account, and so on. For example:

FtpSession1.ServerName=ftp.com

5. Call session methods, such as Connect to Host, User Login, Get, and so on. For example:

FtpSession1.UserLogin

Related Topics

Introducing FTP OLE API Sample Client Source Code

IHclFtpEngine Object

The IHclFtpEngine object provides information on the current application such as its name, parent and the state of the main window. Create the IHclFtpEngine object before you create any other object.

The methods and properties of this object are for use with this object and for versions 6.0 or later only.

Related Topics IHclFtpEngine Object Properties IHclFtpEngine Object Method

IHclFtpSession Object

The IHclFtpSession object includes methods for executing a wide range of commands for managing files, printing, and connecting to the host. The IHclFtpSession object 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

IHclFtpSessions Object

The IHclFtpSessions object creates and manages IHclFtpSession objects (a collection of FTP sessions). The IHclFtpSessions object 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

User-Defined Types

The following are the user-defined types:

- TFirewall Data Type
- TLocalCase Data Type
- TTransfer Data Type

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

<u>Conversing with Telnet using DDE</u> <u>Automating Telnet using OLE</u>

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 Topic Request Items (HTelnet|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 Topic Items (HTelnet|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 Type Topic Items (HTelnet|EmulationType)—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 Items (HTelnet|Page)—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
- Systems Item
- Formats Item

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:

- <u>ConnectionStatus Item</u>
- <u>EmulationType Item</u>
- ExecReturn Item
- Formats Item
- <u>SaveData Item</u>
- 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
- <u>ConnectRlogin Item</u>
- <u>ConnectTelnet Item</u>
- <u>Disconnect Item</u>
- <u>SendString Item</u>
- <u>SendStringToTerminal Item</u>
- LookForString Item
- StartDataSave Item
- StopDataSave Item

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:

- <u>TopicItemList Item</u>
- Formats Item
- <u>CursorPosition Item</u>
- 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
- <u>SetOption Item</u>
- ReadKeyboardFile Item
- <u>SetAttributeColor Item</u>
- <u>SetAttributeUsage Item</u>

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:

- <u>TopicItemList Item</u>
- Formats Item
- <u>CurrentFGColor Item</u>
- <u>CurrentBGColor Item</u>
- <u>CurrentAttribute Item</u>
- <u>CharacterAt Item</u>
- <u>CharacterAttributeAt Item</u>
- <u>CharacterFGColorAt Item</u>
- <u>CharacterBGColorAt Item</u>

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:
 - WyseTerm Object
 - Emulation Object
 - Page Object
- 3. Specify the appropriate <u>Property and Method Syntax</u> corresponding to the object you want to use.

Related Topics

Starting Telnet in Hummingbird Basic Using OLE Available OLE Objects in Telnet

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 WyseTerm object. For example,

set Telnet=CreateObject("Hummingbird.WyseTerm")

If the application is already running, connect to it using the following statement:

```
set Telnet=GetObject(,"Hummingbird.WyseTerm")
```

Note: Note the comma preceding "Hummingbird Telnet".

3. Configure Telnet using the Property and Method Syntax included.

Related Topics

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:

```
WyseTerm.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=WyseTerm.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 WyseTerm object's method with value as its parameter:

WyseTerm.*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:

WyseTerm 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 WyseTerm 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 WyseTerm Object as the LPDISPATCH pageObject property. Use this object to control all information related to the screen and its contents.

<u>**Title Object</u>**—The Title object is referenced through the WyseTerm Object as the LPDISPATCH title property. This property holds all information related to the title and its contents, and is both readable and writable.</u>

Note: The Emulation object and the Page objects are always referenced through the WyseTerm object

WyseTerm Object

The WyseTerm 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 WyseTerm object has the creatable name of hummingbird.WyseTerm, and consists of the following properties and methods.

Properties

- EmulationType Property
- <u>EmulationObject Property</u>
- Title Object

- ConnectionStatus Property
 - Page Object

Methods

- LoadSetupFile Method
- ConnectTelnet Method
- Visible BOOL Method
- Fullname Method
- SendStringToTerminal Method
- <u>GetEventStatus Method</u>
- <u>StartDataSave Method</u>
- StopDataSave Method

- ConnectRlogin Method
- Disconnect Method
- Name Method
- SendString Method
- LookForString Method
- RemoveEvent Method
- ReadData Method

Emulation Object

The Emulation object is referenced through the WyseTerm 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

Page Object

The Page object is referenced through the WyseTerm 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 Property
- BGColor Property
- Attribute Property

Methods

- GetCharacter Method
- <u>GetCharacterAttrib Method</u>
- GetCharacterBGColor Method
- SetCharacterAttrib Method
- <u>SetCharacterBGColor Method</u>
- GetCharacterString Method
- <u>GetCharacterFGColor Method</u>
- SetCharacter Method
- <u>SetCharacterFGColor Method</u>

Title Object

The Title object is referenced through the WyseTerm Object as the LPDISPATCH title property. This property holds all information related to the title and its contents, and is both readable and writable.

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.WyseTerm")
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
```

General Accessibility

Hummingbird products are accessible to all users. Wherever possible, our software adheres to Microsoft Windows interface standards and contains a comprehensive set of accessibility features.

Access Keys All menus have associated access keys (mnemonics) that let you use the keyboard, rather than a mouse, to navigate the user interface (UI). These access keys appear as underlined letters in the names of most UI items. (If this is not the case, press Alt to reveal them.) To open any menu, press Alt and then press the key that corresponds with the underlined letter in the menu name. For example, to access the File menu in any Hummingbird application, press Alt+F.

Once you have opened a menu, you can access an item on the menu by pressing the underlined letter in the menu item name, or you can use the arrow keys to navigate the menu list.

Keyboard Shortcuts Some often-used menu options also have shortcut (accelerator) keys. The shortcut key for an item appears beside it on the menu.

Directional Arrows Use the directional arrows on the keyboard to navigate through menu items or to scroll vertically and horizontally. You can also use the directional arrows to navigate through multiple options. For example, if you have a series of radio buttons, you can use the arrow keys to navigate the possible selections.

Tab Key Sequence To navigate through a dialog box, press the Tab key. Selected items appear with a dotted border. You can also press Shift+Tab to go back to a previous selection within the dialog box.

Spacebar Press the Spacebar to select or clear check boxes, or to select buttons in a dialog box.

Esc Press the Esc key to close a dialog box without implementing any new settings.

Enter Press the Enter key to select the highlighted item or to close a dialog box and apply the new settings. You can also press the Enter key to close all About boxes.

ToolTips ToolTips appear for all functional icons. This feature lets users use Screen Reviewers to make interface information available through synthesized speech or through a refreshable Braille display.

Microsoft Accessibility Options

Microsoft Windows environments contain accessibility options that let you change how you interact with the software. These options can add sound, increase the magnification, and create sticky keys.

To enable/disable Accessibility options:

- 1. In Control Panel, double-click Accessibility Options.
- 2. In the Accessibility Options dialog box, select or clear the option check boxes on the various tabs as required, and click Apply.
- 3. Click OK.

If you installed the Microsoft Accessibility components for your Windows system, you can find additional accessibility tools under Accessibility on the Start menu.

Technical Support

You can contact the Hummingbird Technical Support department Monday to Friday between 8:00 a.m. and 8:00 p.m. Eastern Time.

Hummingbird Ltd. 1 Sparks Avenue, Toronto, Ontario, Canada M2H 2W1		
	Canada and the USA	International
Technical Support:	1-800-486-0095	+1-416-496-2200
General Enquiry:	1-877-FLY-HUMM	+1-416-496-2200
Main:	+1-416-496-2200	
Fax:	+1-416-496-2207	
E-mail:	<pre>support@hummingbird.com</pre>	
FTP:	<u>ftp.hummingbird.com</u>	
Web Support:	<pre>support.hummingbird.com/customer</pre>	
Web Site:	www.hummingbird.com	

Cfg3270 Objects List

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
```

Cfg5250 Objects List

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.

```
'$include:"-E\hebasic.ebh"
Example
              Sub Main
                Dim HE as Object
                Set HE = CreateObject( "HostExplorer" )
                HE.CurrentHost.Cfg5250.CursorType = CURSOR_BLOCK
               End Sub
               '$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
```

CfgVT Objects List

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("HostEx lorer")	
	HE.CurrentHost.CfgVT.CursorTy e = CURSOR_BLOCK End Sub	
	'\$include:"-E\hebasic.ebh"	
	Sub Main Dim HE as Object Set HE = CreateObject("HostEx lorer")	
	With HE.CurrentHost.CfgVT.CursorTy e = CURSOR_BLOCK End With End Sub	
	'\$include:"-E\hebasic.ebh"	
	Sub Main Dim Cfg as Object Dim HE as Object Set HE = CreateObject("HostEx lorer") Set Cfg = HE.CurrentHost.CfgVT	
	Cfg.CursorTy e = CURSOR_BLOCK End Sub	

Assignment Group

3270 5250 VT

Erase—Reinitializes the elements of fixed-size arrays and frees dynamic storage space. Not supported by Hummingbird Basic. Supported by Microsoft Visual Basic.

Let—Assigns 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 userdefined type to another variable of a different user-defined type.

Set—Assigns an object reference to a variable or property.

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.

Conversion Group

3270 5250 VT

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.

DDE Group

3270 5250 VT

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.

DefType Statement

3270 5250 VT

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.

Error Handling Group

3270 5250 VT

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.

Flow Control Group

3270 5250 VT

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.

Miscellaneous Group

3270 5250 VT

'—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.

Operators Group

3270 5250 VT

^—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.

String Group

3270 5250 VT

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.

User Dialog Group

3270 5250 VT

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.

Variable Info Group

3270 5250 VT

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.

Constant Group

3270 5250 VT

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.

Data Type Group

3270 5250 VT

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.

Declaration Group

3270 5250 VT

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 applicationlevel 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.

Dialog Methods and Statements

3270 5250 VT

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.

DIgControlld—Returns the numeric ID of a dialog box control.

DIgCount—Returns the number of dialog box items in the dialog box. Not supported by Hummingbird Basic.

DigEnable (method)—Indicates whether a control is enabled or disabled.

DIgEnable (statement)—Enables or disables a dialog box control.

DIgEnd—Closes the dialog box.

DIgFocus—Sets the focus to a dialog box control.

DIgListBoxArray (method)—Returns the contents of a list box or combo box.

DIgListBoxArray (statement)—Sets the contents of a list box or combo box.

DIgName—Returns the field name. Not supported by Hummingbird Basic.

DIgNumber—Returns the number of the expression that returns a string result. Not supported by Hummingbird Basic.

DIgSetPicture—Changes the picture in a picture dialog box control for the current dialog box. Not supported by Hummingbird Basic.

DIgText (method)—Returns the text associated with a dialog box control.

DIgText (statement)—Sets the text associated with a dialog box control.

DIgType—Returns a string value indicating the type of expression that returns a numeric result. Not supported by Hummingbird Basic.

DIgValue (method)—Returns the value associated with a dialog box control.

DIgValue (statement)—Sets the value associated with a dialog box control.

DIgVisible (method)—Indicates whether a control is enabled or disabled.

DlgVisible (statement)—Shows or hides a dialog box control.

File Group

3270 5250 VT

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.

Math Group

3270 5250 VT

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.

Object Group

3270 5250 VT

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.

Settings Group

3270 5250 VT

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.

TimeDate Group

3270 5250 VT

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 Input Group

3270 5250 VT

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.

Entering Control Sequences

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

Renamed Interfaces of the Profile Object

The following interfaces of the Profile object have been renamed:

Former Name	New Name
ProfileEditing	ProfileEdit
ProfilePrintExplorer	ProfilePrintSession
ProfileSaveFile	ProfileCapture
ProfileVTPrint	ProfileHostPrinting

Related Topics

Renamed or Moved COM Methods, Properties, and Interfaces

Renamed Properties of the Profile Object

The following properties of the Profile object have been renamed:

Former Name	New Name	Interface
TraceFilename	HLLAPITraceFilename	Profile
EnableEmuTracing	AllowEmuTracing	Profile
Editing	Edit	Profile
PrintExplorer	PrintSession	Profile
VTPrint	HostPrinting	Profile
SaveFile	Capture	Profile
SpecialModel	CustomModel	ProfileTerminal
SpecialModelCols	CustomModelCols	ProfileTerminal
SpecialModelRows	CustomModelRows	ProfileTerminal
VTMoveCursorOn MouseClick	MoveCursorOn MouseClick	ProfileCursor
UseSpecificPrinter	PRTSCRUseSpecific Printer	ProfilePrintScreen
PEHostName	HostName	ProfilePrintSession
PELUName	LUName	ProfilePrintSession
PELUType	LUType	ProfilePrintSession
PEProfileName	ProfileName	ProfilePrintSession
PEStartPrinter	StartPrinter	ProfilePrintSession
PEStopPrinter	StopPrinter	ProfilePrintSession
ShowRecvDir	ShowRecvDlg	ProfileFileTransfer
7171PrintMode	PrintMode7171	ProfilePCPrint

Related Topics Renamed or Moved COM Methods, Properties, and Interfaces

Moved Properties of the Profile Object

The following properties have been moved from one interface to another:

Property	Former Interface	New Interface
AlternateScreen	Profile	ProfileTerminal
AreaCode	Profile	ProfileConnection
AutoMacroName	Profile	ProfileConnection
CharacterSpacing	Profile	ProfileFonts
ConnectBy	Profile	ProfileConnection
CountryCode	Profile	ProfileConnection
CountryID	Profile	ProfileConnection
DeviceName	Profile	ProfileConnection
DirectToModem	Profile	ProfileConnection
FullScreenMode	Profile	ProfileSessionWindow
HostName	Profile	ProfileConnection
Port	Profile	ProfileConnection
PromptOnClose	Profile	ProfileSessionWindow
SaveProfOnClose	Profile	ProfileSessionWindow
Schemes	Profile	ProfileEvents, ProfileColor, ProfileFileTransfer, ProfileHotspots
ShowDialupDlg	Profile	ProfileConnection
SYSREQasIACIP	Profile	ProfileConnection
TelnetEcho	Profile	ProfileConnection
TelnetName	Profile	ProfileConnection

Timeout	Profile	ProfileConnection
TypeAhead	Profile	ProfileKeyboard
TypeAheadTimeout	Profile	ProfileKeyboard
UponDisconnect	Profile	ProfileConnection
UseDialProp	Profile	ProfileConnection
UserName	Profile	ProfileConnection
VariableWidthFont	Profile	ProfileFonts
LongName	Profile	ProfileSessionWindow
LUName	Profile	ProfileConnection
ModemID	Profile	ProfileConnection
Notify	Profile	ProfileSound
Password	Profile	ProfileConnection
VTDoHostWindow Size	Profile	ProfileConnection
VTInitiateTelnet Negotiation	Profile	ProfileConnection
VTLineMode	Profile	ProfileConnection
Sound	Profile	ProfileSound
WindowTitle	Profile	ProfileSessionWindow
AutoRunMacro DelayTime	Profile	ProfileConnection
ColumnSeparators	ProfileTerminal	ProfileDisplay
VTDefColsPerScreen	ProfileDisplay	ProfileTerminal
VTDefLinesPerScreen	ProfileDisplay	ProfileTerminal
NoLockKeyb	ProfileDisplay	ProfileEdit
MultiLineDelete	ProfileDisplay	ProfileEdit

	<u> </u>	<u> </u>
MultiLineInsert	ProfileDisplay	ProfileEdit
InsertResetByAttn	ProfileDisplay	ProfileEdit
RespectNumeric	ProfileDisplay	ProfileEdit
ConvertNulls	ProfileDisplay	ProfileEdit
AlwaysAutoskip	ProfileDisplay	ProfileEdit
EnableWorkspace BackgroundBitmap	ProfileDisplay	ProfileSessionWindow
WorkSpace BackgroundBitmap	ProfileDisplay	ProfileSessionWindow
WorkSpace BackgroundColor	ProfileDisplay	ProfileSessionWindow
WorkSpace ForegroundColor	ProfileDisplay	ProfileSessionWindow
DisplayBorder	ProfileFonts	ProfileSessionWindow
SaveFontOnExit	ProfileFonts	ProfileSessionWindow
SnapFrameBack	ProfileFonts	ProfileSessionWindow
SwitchScreenType	ProfileFonts	ProfileSessionWindow

Related Topics

Renamed or Moved COM Methods, Properties, and Interfaces

HEPARSER_FEATURE Data Type

The HEPARSER_FEATURE data type indicates the type of feature that is enabled or disabled.

It has the following values:

Value	Definition
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_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 and applies only to TN5250 terminals.
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_CAPTURE_SCREEN	Determines whether HostExplorer saves all information on the screen. By default, this data type value is FALSE and applies only to TNVT terminals.
HOSTEX_CLIPBOARD_FORMAT_BITMAP	Determines whether HostExplorer enables bitmap format when copying data to the Clipboard. By default, this data type value is TRUE.

	<u> </u>
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 when 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_LOCK_ON_ATTENTION	Determines whether HostExplorer locks the keyboard after you send an attention key command. By default, this value is FALSE.
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_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 TN3270 and TN5250 terminals.

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:

Value	Definition
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 AttentionFormat AttnFormat

HOSTEX_BACKSPACE_KEY_INTERPRETATION Data Type

3270 5250

The HOSTEX_BACKSPACE_KEY_INTERPRETATION data type specifies what command HostExplorer sends to the host when you press the Backspace key.

Value	Definition
HOSTEX_BACKSPACE_KEY_AS_DELETE	Specifies that HostExplorer sends the Delete command to the host every time you press the Backspace key. The Delete command deletes the character to the immediate right of the cursor.
HOSTEX_BACKSPACE_KEY_AS_BACKSPACE	Specifies that HostExplorer sends the Backspace command to the host every time you press the Backspace key. The Backspace command deletes the character to the immediate left of the cursor.

Related Topics BackspaceKeyInterpretation

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:

Value	Definition
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 CellCopyMode CellDelimited

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:

Value	Definition
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 <u>ConnectBy (IHEParser)</u> <u>ConnectBy (IHEProfileConnection)</u> <u>ConnectBy (IHETerminal)</u>

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:

Value	Definition
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

HOSTEX_ENTER_KEY_INTERPRETATION Data Type

3270 5250

The HOSTEX_ENTER_KEY_INTERPRETATION data type specifies what key sequence HostExplorer sends to the host when you press the Enter key.

Value	Definition
HOSTEX_ENTER_KEY_AS_RETURN_AND_LINEFEED	Specifies that HostExplorer sends the key sequence CR + LF (carriage return + linefeed) every time you press Enter.
HOSTEX_ENTER_KEY_AS_CARRIAGE_RETURN	Specifies that HostExplorer sends the key sequence CR (carriage return) every time you press Enter.

Related Topics EnterKeyInterpretation

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:

Value	Definition
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_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:

Value	Definition
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

PSCellSize

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:

Value	Definition
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

<u>GraphicsCursorType</u>

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:

Value	Definition
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 GraphicsModel (IHEParser) GraphicsModel (IHEProfileGraphics)

HOSTEX_HOTSPOT_DISPLAY Data Type

3270 5250 VT

The HOSTEX_HOTSPOT_DISPLAY data type specifies how HostExplorer displays hotspots.

It has the following values:

Value	Definition
HOSTEX_HOTSPOT_DISPLAY_INVISIBLE	Specifies that each hotspot text or region appears in its regular display style (not highlighted) until you place your cursor over it, at which point the cursor turns into a hand to indicate the presence of the hotspot.
HOSTEX_HOTSPOT_DISPLAY_RAISED_BUTTON	Specifies that each hotspot text or region appears highlighted.

Related Topics

<u>DisplayStyle</u>

HOSTEX_HOTSPOT_MOUSE_ACTIVATION Data Type

3270 5250 VT

The HOSTEX_HOTSPOT_MOUSE_ACTIVATION data type specifies how hotspots are activated using the mouse.

It has the following values:

Value	Definition
HOSTEX_HOTSPOT_MOUSE_ACTIVATION_SINGLE_CLICK	Specifies that hotspots activate when you click them once with the left mouse button.
HOSTEX_HOTSPOT_MOUSE_ACTIVATION_DOUBLE_CLICK	Specifies that hotspots activate when you click them twice with the left mouse button.

Related Topics

MouseActivation

HOSTEX_KEYBOARD_BUFFER_MODE Data Type 3270 5250

The HOSTEX_KEYBOARD_BUFFER_MODE data type specifies how HostExplorer stores characters in a buffer until they are sent to the host.

It has the following values:

Value	Definition
HOSTEX_KEYBOARD_BUFFER_AS_CHARACTER_MODE	Specifies that HostExplorer sends each character immediately to the host.
HOSTEX_KEYBOARD_BUFFER_AS_LINE_MODE	Specifies that HostExplorer sends characters one line at a time until you press the Enter key.

Related Topics

KeyboardBufferMode

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

Value	Definition
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.
HOSTEX_KEYBOARD_TYPE_PC_105	Indicates that the PC keyboard has 105 keys.

Related Topics KeyboardType

HOSTEX_LINEMODE Data Type

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.

Value	Definition
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.
HOSTEX_LINEMODE_RFCOMPLIANT	Enables compliance with Telnet RFC specifications.

It has the following values:

Related Topics

VTLineMode

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:

Value	Definition
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

OnPasteFieldModeTabCharacter HOSTEX_PASTE_MODE Data Type

HOSTEX_OIA_DISPLAY Data Type

3270 5250

The HOSTEX_OIA_DISPLAY data type specifies whether HostExplorer displays the host IP address or the host response time in the OIA (Operator Information Area).

It has the following values:

Value	Definition
HOSTEX_OIA_DISPLAY_IP_ADDRESS	Specifies that HostExplorer displays the host IP address in the OIA.
HOSTEX_OIA_DISPLAY_HOST_RESPONSE_TIME	Specifies that HostExplorer displays the host response time in the OIA.

Related Topics DisplayInOIA

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:

Value	Definition
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_	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

PasteMode HOSTEX_NEXT_FIELD_KEY Data Type

HOSTEX_PRINT_TARGET Data Type

The HOSTEX_PRINT_TARGET data type specifies the target for host printing. It has the following values:

Value	Definition
HOSTEX_PRINT_TARGET_DEFAULT_PRT	Specifies that HostExplorer uses the default printer for host print jobs.
HOSTEX_PRINT_TARGET_SPECIFIC_PRT	Specifies that HostExplorer uses a specified printer for host printing.
HOSTEX_PRINT_TARGET_FILE	Specifies that HostExplorer prints host print jobs to a specified file.

Related Topics

VTPrintTarget

VTPrintFile

VTUseSpecificPrinter

HOSTEX_PRINTFILE_MODE Data Type

The HOSTEX_PRINTFILE_MODE data type specifies the method that HostExplorer uses to print to a file.

The data type has the following values:

Value	Definition
HOSTEX_PRINTFILE_MODE_OVERWRITE	Specifies that HostExplorer overwrites the target file with the new material.
HOSTEX_PRINTFILE_MODE_APPEND	Specifies that HostExplorer appends the new material to the existing contents of the target file.
HOSTEX_PRINTFILE_MODE_AUTO_NUMBER	Specifies that HostExplorer prefixes a line number to each line it writes to the target file.

Related Topics VTPrintFileMode

HOSTEX_RESIZE_BEHAVIOR Data Type

The HOSTEX_RESIZE_BEHAVIOR data type specifies how HostExplorer displays information in the session window when you resize the window.

Value	Definition
HOSTEX_RESIZE_BEHAVIOR_CHANGE_FONT	Specifies that HostExplorer changes the size of the font to allow the same number of rows and columns to be displayed in the resized window.
HOSTEX_RESIZE_BEHAVIOR_NEGOTIATE_WIN_SIZE	Specifies that HostExplorer sends a change in the number of maximum rows and columns to the Telnet host when you resize the window, but does not change the font size. This value is valid only for Telnet hosts that support the NAWS (Negotiate About Window Size) option.
HOSTEX_RESIZE_BEHAVIOR_DO_NOTHING	Specifies that HostExplorer does not do anything when you resize the window.

Related Topics ResizeBehavior

HOSTEX_SAVE_OPTIONS Data Type

3270 5250 VT

The HOSTEX_SAVE_OPTIONS data type specifies the components of the Profile object that you can save.

It has the following values:

Value	Definition
HOSTEX_SAVE_ALL	Specifies that HostExplorer saves the values of all components in the Profile object.
HOSTEX_SAVE_FONTS	Specifies that HostExplorer saves any changes made to the session font.
HOSTEX_SAVE_EVENT_SCHEME	Specifies that HostExplorer saves the event scheme for the current session.

Related Topics

<u>Save</u>

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:

Value	Definition
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 between 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 Security Option (IHETransport) Security Option (IHEProfileSecurity)

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:

Value	Definition
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 SelectionMode

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:

Value	Definition
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.
HOSTEX_STATUS_LINE_MODE_5250TERMIMALSTATUSBAR	Indicates that HostExplorer displays a 5250-terminal-style status bar.

Related Topics

<u>StatusLineMode (IHEParser)</u> <u>StatusLineMode (IHEProfileDisplay)</u>

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:

Value	Definition
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

<u>SwitchScreenType</u>

HOSTEX_TELNETECHO Data Type

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:

Value	Definition
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 <u>TelnetEcho (IHETransport)</u> <u>TelnetEcho (IHEProfileConnection)</u>

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:

Value	Definition
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

<u>TprintMode</u>

Properties and Data Types of the HEPAR3270 Object

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

Properties

- APLInputMode
- GraphicsModel
- PrinterDeInitString
- ProgramSymbols
- TransferMode
- EnableAPL
- NumericCharacters
- PrinterInitString
- TransferErrorCode
- <u>ValidateNumericFieldData</u>

Data Types

- HOSTEX_GRAPHICS_CURSOR_TYPE Data Type
- HOSTEX_GRAPHICS_MODEL Data Type
- HOSTEX_INSERT_KEY_STYLE Data Type

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

- GetFieldAttribute
- GetFieldExtAttribute
- GetFieldPos
- GetScreenText
- IsFieldBold
- IsFieldModified
- IsFieldPenSelectable
- PasteDataToScreen
- SetFieldText
- WriteProtectedText

- GetFieldCount
- GetFieldIndex
- GetFieldText
- HostResponseTime
- IsFieldHidden
- IsFieldNumeric
- IsFieldProtected
- PutString
- WaitForIO

Properties

- ConvertNulls
- <u>OnCopyReplaceFieldAttributeWith</u>
- PasteMode

- CutMode
- <u>OnPasteFieldModeTabCharacter</u>

Data Types

- HOSTEX_CELL_DELIMITED Data Type
- HOSTEX_FIELD_ATTR_REPLACEMENT Data
 Type
- HOSTEX_PASTE_MODE Data Type
- HOSTEX_CUT_MODE Data Type
- HOSTEX_NEXT_FIELD_KEY Data Type
- HOSTEX_STATUS_LINE_MODE Data
 Type

Methods of the HEPAR3270/HEPARVT Objects

The following methods are specific to the HEPAR3270 and HEPARVT objects:

- SendFile
- ReceiveFile

Properties and Data Types of the HEPARVT Object

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

Properties

- Answerback
- BufRows
- EnablePrinterTimeout
- <u>HostWritableString</u>
- NRCID
- PrintDisableTranslation
- PrintLFtoCRLF
- <u>SaveFileName</u>
- TerminalID

- AutoWrap
- CaptureMode
- HistoryLines
- MoveCursorOnMouseClick
- PrintByPassWindows
- PrinterTimeoutValue
- SaveAppend
- SoftCharacterSetID
- UPSS

Data Types

- HOSTEX_CAPTURE_MODE Data Type
- HOSTEX_TERMINAL_ID Data Type

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

- AsciiToHost
- FindString
- GetFeature
- GetSel
- GetValue
- MoveCursorRelative
- ReplaceSel
- SendKeys
- SetFeature
- SetValue
- WaitForCursor
- WaitForString
- WaitIdle
- WaitXfer

- ClearSel
- GetCursorPosition
- GetFieldLength
- GetSelectionArea
- HostToAscii
- PutText
- SendAid
- SetCursorPosition
- SetSel
- WaitConnected
- WaitForCursorMove
- WaitHostQuiet
- WaitPSUpdated

Properties

- CanChangeScreen
- ConnectBy
- ConnectRC
- KeyboardLocked
- ModelRows
- OIAString
- ScreenChanged
- ScreenRow
- SessionName
- TerminalModel
- Transport

- CellCopyMode
- ConnectErrorStatus
- HLLAPIName
- ModelColumns
- NVTMode
- OIAStringW
- ScreenCol
- SelectionMode
- StatusLineMode
- Text
- TypeAheadTimeout

Data Types

- <u>HEPARSER_FEATURE Data Type</u>
- HOSTEX_CONNECT_BY Data Type
- HOSTEX_SELECTION_MODE Data Type
- HOSTEX_TERM_MODEL Data Type

HOSTEX_CAPTURE_MODE Data Type

The HOSTEX_CAPTURE_MODE data type specifies how to capture selected text.

It has the following values:

Value	Definition
HOSTEX_CAPTURE_MODE_RAW	Indicates that the system captures all data, including escape sequences, received by the emulator.
HOSTEX_CAPTURE_MODE_TEXT	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

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:

Value	Definition
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.

HOSTEX_TERMINAL_ID Data Type

The HOSTEX_TERMINAL_ID data type specifies the terminal ID response that HostExplorer sends to the host.

It has the following values:

Value	Definition
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

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:

Value	Definition
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_VT52	Indicates that you are using a VT52 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_	Indicates that you are using an IBM 3151 terminal to connect to the host.
HOSTEX_TERM_MODEL_WYSE50	Indicates that you are using a WYSE50 terminal to connect to the host.
HOSTEX_TERM_MODEL_WYSE60	Indicates that you are using a WYSE60 terminal to connect to the host.

Related Topics

<u>TerminalModel</u>

HOSTEX_TRANSFER Data Type

3270

The HOSTEX_TRANSFER data type specifies whether to download or upload files.

It has the following values:

Value	Definition
HOSTEX_TRANSFER_DOWNLOAD	Indicates that HostExplorer downloads files.
HOSTEX_TRANSFER_UPLOAD	Indicates that HostExplorer uploads files.

Related Topics TransferMode

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:

- LUNameRequested
- Keyboard
 MessageQueueName
- MessageQueueLibrary
- Username
- Password
- ∎ <u>Userna</u>

Properties of the HETP3270/5250 Objects

The following properties are specific to both the HETP3270 and the HETP5250 objects:

- DeviceType
- LUNameRequested

Properties and Data Types of the HETPVT Object

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

Properties

- LineMode
- <u>TerminalOnline</u>

Data Types

- HOSTEX_LINEMODE Data Type
- HOSTEX_TELNETECHO Data Type

Properties of the HETP3270/VT Objects

The following properties are specific to both the HETP3270 and the HETPVT objects:

- EnableSSH
- IsEncrypted

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
- CodePage
- ConnectionStatus
- HostAddress
- IsReceiveBlocked
- ModelRows
- Port
- RetryDelayTimeBetweenHosts
- TelnetIsLineMode
- TelnetName
- TerminalType

- CharSet
- Connected
- EnableTracing
- HostName
- ModelColumns
- NumberOfRetries
- PortList
- TelnetEcho
- TelnetIsLocalEcho
- TerminalModel
- TraceFilename

Data Types

- HOSTEX_ATN_FORMAT Data Type
- HOSTEX_CONNECT_BY Data Type
- HOSTEX_DEVICE_TYPE Data Type
- HOSTEX_FUNCTION_KEY Data Type
- HOSTEX_TERM_MODEL Data Type
- HOSTEX_TOGGLE_RECEIVE Data Type

HETRANSPORT_FEATURE Data Type

3270 5250 VT

The HETRANSPORT_FEATURE data type enumerates the configurable features of the Transport objects. It has the following values:

Value	Definition
HOSTEX_E_MODE	Enables/disables 3270 E or 5250 E mode.
HOSTEX_EAB	Enables/disables the Extended Attribute feature (3270 only).
HOSTEX_INITIATE_TELNET_NEGOTIATION	Enables/disables Telnet negotiation (VT only).
HOSTEX_ENABLEKERBEROSTICKETFORWARDING	Enables/disables Kerberos ticket forwarding (3270 and VT only).
HOSTEX_ENABLEKERBEROSAUTHENTICATION	Enables/disables Kerberos authentication (3270 and VT only).
HOSTEX_ENABLEKERBEROSENCRYPTION	Enables/disables Kerberos encryption (3270 and VT only).
HOSTEX_ENABLESSLTLSUSERCERTIFICATE	Enables/disables the user certificate for SSL authentication.
HOSTEX_ENABLESSLTLSSERVERCERTIFICATE	Enables/disables the server certificate for SSL authentication.
HOSTEX_ENABLESSLTLSREQUESTCERTIFICATE	Enables/disables the SSL request certificate feature.
HOSTEX_ENABLESSLTLSNEGOTIATIONFAILURE	Enables/disables SSL negotiation failure.
HOSTEX_ENABLESSLTLSNEGOTIATEVIATELNET	Enables/disables SSL negotiation over Telnet.
HOSTEX_ENABLESSLTLSSECURITY	Enables/disables the SSL security feature.

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:

Value	Definition
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 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:

Value	Definition
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

<u>DeviceType</u>

HOSTEX_ENCRYPTED Data Type

3270 5250 VT

The HOSTEX_ENCRYPTED data type specifies the encryption level of the session.

It has the following values:

Value	Definition
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

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:

Value	Definition
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 SendFunctionKey

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:

Value	Definition
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 ToggleBlockReceive

OHIO_DIRECTION Data Type

The OHIO_DIRECTION data type specifies the direction of the search.

It has the following values:

Value	Description
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 FindByString FindString

OHIO_STATE Data Type

The OHIO_STATE data type specifies the status of the communication link to the host.

It has the following values:

Value	Description
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 OnSessionChanged

OHIO_TYPE Data Type

The OHIO_TYPE data type specifies the type of host.

It has the following values:

Value	Description
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 SessionType

OHIO_COLOR Data Type

The OHIO_COLOR data type specifies the color of the text in the entire field.

It has the following values:

Value	Descripton
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 GetData (IHEOhioField) GetData (IHEOhioScreen)

OHIO_EXTENDED Data Type

The OHIO_EXTENDED data type specifies the extended attribute of the field.

It has the following values:

Value	Description
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 GetData (IHEOhioField) GetData (IHEOhioScreen)

OHIO_FIELD Data Type

The OHIO_FIELD data type specifies the field type.

It has the following values:

Value	Description
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 GetData (IHEOhioScreen)

OHIO_PLANE Data Type

The OHIO_PLANE data type specifies the plane from which to retrieve the data.

It has the following values:

Value	Description
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 GetData (IHEOhioField) GetData (IHEOhioScreen)

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:

Value	Description
OHIO_UPDATE_HOST	Indicates that the host initiated the update.
OHIO_UPDATE_CLIENT	Indicates that the client initiated the update.

Related Topics OnScreenChanged

OHIO_INPUTINHIBITED Data Type

The OHIO_INPUTINHIBITED data type specifies what is inhibiting the input.

It has the following values:

Value	Description
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

OHIO_OWNER Data Type

The OHIO_OWNER data type specifies the owner of the Ohio session.

It has the following values:

Value	Description
OHIO_OWNER_UNKNOWN	Indicates that the owner is uninitialized.
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

<u>Owner</u>

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.

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

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.

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

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.

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

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
```

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

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.

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 Code	Description
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.

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

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	You can poke only data that exists in a cell.
5.0 and 7.0	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 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?

Generic Properties and Methods

The properties and methods listed below are available for all emulation modes.

Properties

- Interpret Property
- <u>BackSpaceDelete Property</u>
- NewLine Property
- LocalEcho Property
- DisplayLines Property
- SmoothScroll Property
- History Property
- CaptureMode Property
- CurrentRow Property

- Autowrap Property
- <u>WarningBell Property</u>
- TermStringId Property
- Online Property
- DisplayWidth80 Property
- JumpScrollNumber Property
- HistorySize Property
- <u>CaptureDestination Property</u>
- <u>CurrentCol Property</u>

Methods

- ReadKeyboardFile Method
- SetAttributeColor Method
- <u>SetAttributeUsage Method</u>

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

- <u>UserPrefCharSetISO Property</u>
- StatusLine Property

WYSE50 and WYSE60 Properties

The properties listed below are available for both the WYSE50 or WySE60 emulation modes:

- <u>ApplicationMode Property</u>
 <u>BlockEndUSCR Property</u>
- AutoPageMode Property
- AutoScroll Property
- ActivePage Property
- CommMode Property

EditModeLocal Property

ActivePageSize Property

OHIO_EXTENDED_HILITE Data Type

The OHIO_EXTENDED_HILITE data type specifies the type of the highlighted text.

It has the following values:

Value	Description
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 GetData (IHEOhioField) GetData (IHEOhioScreen)

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:

Value	Description
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 GetData (IHEOhioField) GetData (IHEOhioScreen)

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. **a**

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

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, and 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

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